AFIT/GST/ENS/92M-05

A METHOD FOR DETERMINING SCHEDULE DELAY INFORMATION IN A CHANNEL CARGO ROUTE NETWORK SCHEDULE

THESIS

Justin Edward Moul
Captain, USAF

AFIT/GST/ENS/92M-05

AFIT/GST/ENS/92M-05

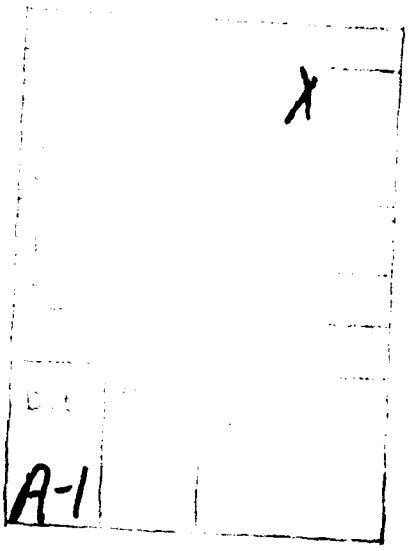
A METHOD FOR DETERMINING SCHEDULE DELAY INFORMATION IN A CHANNEL CARGO ROUTE NETWORK SCHEDULE

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Justin Edward Moul, B.S., M.B.A.
Captain, USAF

June, 1992



THESIS APPROVAL

STUDENT: Captain Justin E. Moul CLASS: GST-92M

THESIS TITLE: A Method for Determining Schedule Delay Information in a Channel Cargo Route Network Schedule

DEFENSE DATE: 24 April 1991

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Co-advisor	Lt Col James T. Moore/ENS	James T. Moore
Co-advisor	Capt John J. Borsi/ENS	John J. Borsi

Acknowledgments

Without the help and support of many people, I could never have completed this project. I would like to take this opportunity to thank those individuals to whom I am particularly indebted.

My thesis advisors, Lieutenant Colonel James T. Moore and Captain John J. Borsi, deserve thanks for taking on the challenge of guiding me through this thesis effort.

Several of my classmates in the Strategic and Tactical Sciences program have given me the emotional support needed to get me this far. Major Garrison H. Flemings has always been their as the friend and role-model I needed. Special thanks go to Captain Jeffery S. Antes, his wife Lori, their son Justin, and their daughter Kristin – they made me part of their family, and I am forever grateful.

Most importantly, I want to thank my wife Su and my daughter Audra for surviving all of life's troubles without me. May all our future be shared as a family.

Justin Edward Moul

Table of Contents

List of Figures

List of Tables

Abstract

This research develops a method for measuring schedule effectiveness by determining the amounts of enroute cargo delay caused by a given aircraft mission schedule. The method is designed to generate information which helps identify flights for which different scheduling might decrease overall cargo delay in the entire network, given that all non-scheduling factors are held constant. The research uses a simplified twelve-airport cargo route network to test the methodology.

A METHOD FOR DETERMINING SCHEDULE DELAY INFORMATION IN A CHANNEL CARGO ROUTE NETWORK SCHEDULE

I. Introduction

Background

The Military Airlift Command (MAC) of the United States Air Force delivers air cargo between various locations throughout the world. To accomplish this mission, MAC attempts to use available aircraft and personnel in the best way possible. Using information about flight times and historical demand for cargo shipment between locations, a standard route structure for cargo flights has been established. This route structure is referred to as the channel cargo route system. This structure allows for direct shipment between heavy-demand destinations and for transshipment through warehouse airports between light-demand destinations. Unfortunately, simply flying constant numbers of missions over the same routes from month to month may not always yield the best allocation of critical flight resources.

For fiscal year 1989, MAC delivered 320,000 tons of cargo over 930 channels connecting 87 countries at a cost of $720 million (20:11-20). For the European theater January 1991 operations, MAC flew 266 flights on 55 separate missions to deliver 10,000 tons of cargo (15).

Because the amount of cargo tonnage shipped between different locations throughout the world varies every month, the process of determining which routes should be used, and how many missions to fly on each route, must be re-accomplished

every month. MAC must be concerned not only with the effective utilization of available physical resources (e.g., money needed to pay for aircraft fuel and availability of aircrew personnel), but also with its ability to deliver cargo in a timely manner.

The office of Force Structure Analysis at Headquarters MAC (HQ MAC/XPYR) is tasked to determine which, and how many, missions must be flown every month to deliver the forecasted cargo demands. Analysts at HQ MAC/XPYR developed a computer model of this channel cargo route system, which is used to simplify the process of determining effective plans to utilize the available flying resources. The complexity of the computer model has increased over time as the need has arisen for it to incorporate more concerns of the real system.

Terminology

The topic of this research refers to various terms for which explicit definitions may help reduce ambiguity.

The term **channel** just mentioned simply refers to "a pair of bases between which MAC must fly either to deliver cargo or to satisfy a frequency of visit, e.g., an embassy, requirement" (1). In other words, a **channel** refers to a two-location set for which MAC provides (not necessarily direct) delivery service on a regularly scheduled basis.

The verb **stop** refers to the act of arriving at an airport. For instance, an airplane might leave one airport and **stop** at another airport before returning to the home base. As a noun, **stop** refers to the physical place *stopped* at.

An aircraft's **home base** is the airport where the aircraft is normally housed when not flying. Aircraft normally begin and end trips from these locations because of the availability of maintenance facilities, aircrews, and so forth.

A **route** is a description of an aircraft's entire journey from departure of the home base until return to the home base. One typical **route** might include departing

from Dover, flying across the Atlantic Ocean, stopping at Ramstein, flying back over the ocean, and returning to Dover.

A **leg** refers to the travel between two points. In the Dover to Ramstein example, the travel from Dover to Ramstein is the first leg, while the return travel is the second leg.

The term **mission** associates a specific route with a specific type of aircraft.

A **flight** refers to one aircraft flying a mission *at a specific point in time.* For example, in a given month a particular mission might have to be flown fifteen times. Each one of the fifteen requirements would be a **flight.**

An airplane is usually referred to by **tail number.** A specific **tail number** may or may not be associated with specific missions. One **tail number** will most likely be used for multiple flights during a month.

Current Procedure

The process MAC uses for determining which missions to fly for each month is essentially a two-stage process as depicted in Figure 1.1. The solid lines show the major relationship between the two stages, while the dotted lines indicate where information, other than the missions chosen, is shared within the model. HQ MAC/XPYR must re-accomplish this two-stage process every month, to determine how to deliver the forecasted cargo demand (15).

In Stage 1, the cargo forecasts, along with the set of possible routes in the MAC system and a database of other information (e.g., flight times between locations, terminal storage capacities, crew rest requirements), are used in the formulation of a linear integer program. Because of the problem's large size, the integrality requirement is relaxed to permit solution through linear programming. The objective of this linear programming relaxation is to minimize the costs of operating the system, subject to the restriction that all cargo be delivered (15). In other words, the mission

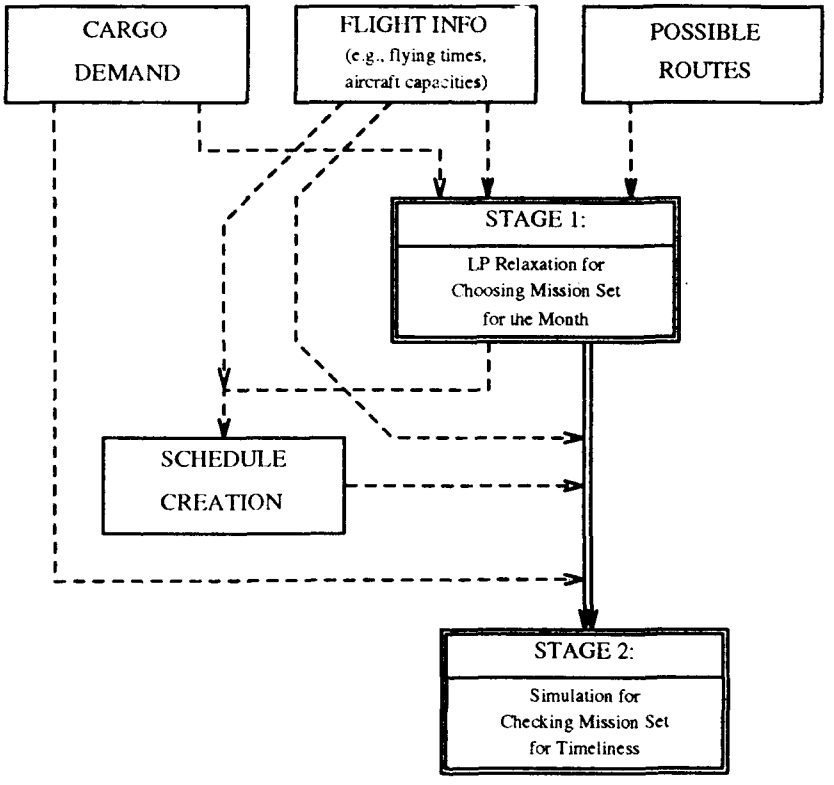


Figure 1.1. The Current MAC Model Procedures

set chosen should provide enough service between all cargo origin-destination (O-D) airports to deliver the forecasted cargo (6).

The results of Stage 1 are known to be approximations, since the linear programming results must be integerized to allow HQ MAC to tell the subordinate operational units how many complete missions must be flown (15). Even though HQ MAC/XPYR does not schedule the actual missions, they must ensure the mission set provided to the operational units can comply with the "Uniform Materiel Movement Issue Priority System (UMMIPS) standards" (19:iii). To ensure compliance with the UMMIPS standards, Stage 2 of HQ MAC/XPYR's process verifies the mission set's ability to deliver the estimated cargo in a timely manner.

As can be seen in Figure 1.1, a schedule of aircraft activity must be created for use in Stage 2 to simulate the actual operations. This schedule might have significant impact on the results obtained from Stage 2. For example, if a bad schedule were created and input into Stage 2, a lack of adherence to the UMMIPS standards might result from the schedule itself — leaving no way to determine whether the mission set is any good. On the other hand, if a good schedule is used for input, the Stage 2 results should not be biased by the schedule — allowing determination of the chosen mission set's ability to adhere to the UMMIPS standards (6).

This verification stage, shown as Stage 2 in Figure 1.1, provides a measure of the delay encountered by cargo being shipped through the MAC network, as expressed in *average delay per cargo ton* shipped between each O-D pair. This delay measurement forms the basis for checking how well the chosen mission set provides timely delivery service for the forecasted cargo demand.

Problem Discussion

Simply put, HQ MAC/XPYR's current mission set evaluation process can suffer from the old adage "garbage in, garbage out." No method currently exists to determine whether a given schedule would be "garbage" or not. Since the results

of the entire model depend upon the schedule, the schedule impacts the validity of Stage 2's results.

For this reason, HQ MAC/XPYR's interests lie in finding a way to evaluate mission schedules, once the other cost factors (i.e., which and how many flights to fly on which routes, using which type of available aircraft resources) have been minimized in Stage 1. HQ MAC/XPYR wants development of some procedure which would produce *better* schedules for use in the simulation portion of the model for timeliness evaluation purposes.

To perform this schedule evaluation, a determination must be made about how to differentiate good schedules from bad schedules. Therefore, the issue focuses on the determination of what information is required for evaluating one schedule against another. Even though the verification stage measures delay, questions persist about the definition of delay, the way to measure delay, and the factors which cause delay. Resolution of these questions might provide a way to evaluate schedules and ultimately produce better schedules for input to Stage 2 of HQ MAC/XPYR's process.

The MAC channel cargo system is an extremely large and complex network. MAC operates hundreds of aircargo terminals throughout the world, with literally hundreds of thousands of potential routes. The issues raised in the preceding paragraphs concerning how to measure delay and how to find the time-dependent aspects of a mission schedule where potential exists for delay reduction are just as important for less complicated route systems.

Purpose of the Research

MAC's current mission-determining process does not provide information in terms of enroute delay for determining whether a given schedule is better than other possible schedules. This research uses a simplified route network to develop a method for measuring schedule effectiveness by determining the amounts of enroute cargo

delay caused by a given mission schedule. This method, if used, generates information which helps identify flights for which different scheduling might decrease overall cargo delay in the entire network, given that all non-scheduling factors are held constant.

To accomplish the purpose of this research, the information generated by the methodology must fulfill two existing needs.

1. The need to know the travel history of every piece of cargo while traveling through a network, including the pickup and dropoff times at each stop, as well as which flights the cargo travelled on.
2. The need to know amounts and types of cargo on every flight leg. This information might help identify sensitivity of the system to potential changes or schedule fluctuation.

Overview of Subsequent Chapters

Chapter II provides highlights of current literature applicable to the area of routing and scheduling. The review focuses on problems associated with the evaluation of delay experienced by cargo while being shipped through a route network.

The method outlined for addressing the problem is explained in Chapter III. The beginning of the chapter describes the inherent assumptions allowing the method to focus specifically on the aspects of a route system dependent on the flight times of a schedule. The chapter includes a description of the procedures used to obtain the desired information about delay caused by a given monthly mission schedule. Finally, the chapter describes the similarities and differences of the proposed information gathering method and the procedure utilized within MAC's Stage 2 simulation program.

Chapter IV describes the sample route system developed for testing the proposed method, and the differences between this system and the larger MAC route

network. The results of applying the proposed methodology to the sample route system are discussed in this chapter. Specific attention emphasizes how the information obtained might be used to improve a mission flight schedule.

Finally, Chapter V provides the researcher's conclusions about the issues addressed in the study and recommendations for further research and improvement.

II. Literature Review

This chapter reviews literature applicable to the research problem, focusing on methods for obtaining information useful for measuring schedule effectiveness.

The Importance of Routing and Scheduling

Scheduling and routing problems have caused great concern for a long time. Cæsar had to decide in what order to schedule the conquering of various regions. Marco Polo had to figure out a route to China. In essence, these two historical figures accomplished tasks that are still done today. Research focuses now

not just on how to select routes and make schedules, but also on how to accomplish these functions more effectively and efficiently.

Solomon and Desrosiers provide the following reasons for the need to develop methods for solving routing and scheduling problems in today's environment:

> The effective and efficient management of the distribution of goods or services is becoming increasingly important in both the private and public sectors. A very important segment of many distribution and transportation system costs is associated with the routing and scheduling of vehicles.
>
> Due to the intrinsic complexity of distribution problems involving routing components, the use of mathematical-programming-based models and algorithms is needed when analyzing and solving such problems to permit the realization of cost reduction or profit improvement. (21:1)

With hundreds of flights providing cargo delivery service to hundreds of airports throughout the world every month, MAC's system is certainly a very important public sector distribution system. Finding more effective and efficient techniques for managing these activities has the potential to reduce public outlays and increase customer satisfaction.

Historically, schedulers repeatedly performed this important, difficult, and often time-consuming, mental and manual task. HQ MAC/XPYR, for instance, must go through the entire routing and scheduling process for its entire fleet *every* month. The age has arrived when computers can assume the burden of accomplishing some of the recurring process. The process still involves people, and computer methods must ensure several competing needs of the process are met. Deal points out:

> ... we found that any approach to this planning should involve the development of a tool that would provide continuity in scheduling operations, delivering schedules at a level of quality equal to that exhibited by those that had been manually produced, as well as a "package"for those personnel newly assigned to this task. (8:573)

Researchers have found it difficult to establish all-encompassing mathematical models for scheduling and routing because each problem has unique characteristics. Some specific problems, such as "bulk-cargo ship scheduling" for the U. S. Navy (12:27) and "internal audit scheduling" (10:267-272), convert to mathematical form. Classes of problems exist, such as the "vehicle routing problem with sliding time windows" (11:213-220) or the "demand-responsive transportation system" (14:630-638), for which certain mathematical forms yield a solution. Notwithstanding these research efforts, Bodin points to some problems: "In my opinion, many of the problems described in literature oversimplify the ones that occur in practice" (3:574).

The experience and research of many authors have resulted in solutions to a number of different scheduling problems. Bodin's caution, however, points to the difficulty of finding solution techniques for specific problems — like the MAC problem. In fact, some research has been concentrated simply to determine how to measure schedule effectiveness.

Measures of Effectiveness

Petersen and Taylor describe scheduling problems as having two distinct parts: "schedule evaluation" and "schedule selection" (17:175). An important aspect of their description is a basis upon which to evaluate or measure schedule performance. Just as a college professor uses a scale for assigning grades to student work, an evaluation scale establishes the method for tying schedule evaluation to schedule selection. This measure of schedule effectiveness must be directly tied to the objective of the scheduling problem at hand.

The objective of a scheduling problem typically relates some measure of effectiveness or efficiency to the performance achieved by ordering activities in some particular sequence. This objective usually relates to two types of performance measures. One type of measure focuses on the system's ability to maximize use of available delivery vehicles or to minimize the monetary cost of operating those resources. The other type of measure focuses on the timeliness with which the mission of shipment delivery is accomplished (See (18:70) and (2:5)). Although related to each other, focus on one type of performance measure (at the expense of the other) can lead to significantly different solution approaches for the same scheduling problem.

Many different interpretations of system usage or monetary cost exist. In the "school-bus routing for program scheduling" problem, Bookbinder and Edwards define cost as the distance required to accomplish all the busing required in the schedule (5:79). Presumably, the distance traveled by the buses translates directly into purchase, maintenance and fuel costs. They also note that other researchers have used "the total number of vehicles required" (5:81) as the measure of monetary cost.

Pritsker highlights several ways for determining how well a system operates, particularly through measures of throughput and resource utilization (18:70). Further extension of this idea might include opportunity costs associated with choosing

one set of activities or activity times when another set could have been chosen instead (11:214). Performance measurements like these apply ideally where the problem objective relates to finding a schedule which uses available resources as much as possible or expends the least amount of money to operate the system.

When the system performance measurement related to maximization of usage or minimization of cost is not key, the performance measurement related to timeliness can become more important. HQ MAC/XPYR's scheduling problem is one where problem emphasis needs a measure focusing on this time dependence. The objectives of their linear programming formulation are the minimization of expense and the maximization in the use of available aircraft (15). To minimize the delay caused through flight scheduling, the focus must now address measurement of performance with respect to timeliness (or customer satisfaction).

As with the HQ MAC/XPYR problem, researchers have addressed similar scheduling problems using this different interpretation of the performance objective. One researcher notes, "one of the basic measures of service level in air transportation is schedule delay" (22:16). Cargo experiences this schedule delay either while flying from airport to airport or while actually sitting on the ground somewhere waiting for further transportation. Although commonly mentioned as important, researchers have not adopted a universally-accepted measurement scale for identifying or evaluating this cargo delay. The literature is replete with ways to measure factors like *lateness*, *tardiness*, and *flowtime* (2, 4, 18). Teodoric mentions that the literature provides various formulas for particular factors (22:16). These formulas can provide statistics for rating system performance in time-dependent terms.

Care must be taken when interpreting results reported in these statistical terms. For instance, although Baker suggests minimizing *mean* or *weighted mean* tardiness or flowtime (2:17-29), the translation to specific event or activity changes to attain better results may not be achievable. When relying on these averaged results, there might not be a realistic way to figure out how individual parts have

affected the overall system performance. A report of average enroute cargo delay may not help identify specific timing changes which might result in better flight schedules, for instance.

For this reason, some problems must be approached with the objectives to make system performance better and to fully capture the activities of each individual element of the system. Byong-Hun and Jae-Yeong state: "Our objective is ... minimum total travel time" (7:394). Dobson and Karmarkar "minimize the total weighted flow time" (9:593). These research teams do not want to bias the results by the use of statistical averages. Measuring performance based on average values might overlook or downplay the importance of unusual occurrences in situations where non-uniform performance within the system is common. With the variability inherent in the MAC channel cargo route model -- particularly resulting from magnitude differences in amounts of cargo flowing between O-D pairs — the schedule optimization needs to address the impact of even the very small subsets of cargo demand which could easily get "lost in the shuffle."

The delay experienced by cargo traveling through the MAC channel route system can be split into two parts. The first part is the time required before initial placement on aircraft at the origin and the time required to unload the aircraft and deliver at the destination. The other part, and the focus of this research, is the time between initial pickup and final dropoff. The scale for measuring schedule effectiveness for this research includes all flying time and ground waiting time as enroute cargo delay (6). Due to the need to adhere to the UMMIPS standards, less enroute delay indicates a better schedule for the MAC problem.

Obtaining Effectiveness Values.

A variety of useful techniques for obtaining better and/or optimal solutions to scheduling problems are described in the literature (See (2, 13, 16)). Many of these techniques attempt to optimize the value of some objective function which

incorporates the effectiveness measurement of scheduling alternatives. For example, one might wish to optimize a problem like Equation (2.1).

$$\min \alpha x + \beta y \tag{2.1}$$

Obviously, for a real (as opposed to theoretic) problem, values must be assigned to the α and β coefficients. These coefficients typically relate to the amount of effectiveness associated with incorporating the different variables (i.e., x and y) in the solution. Many of the optimization techniques, although useful for solving problems like Equation (2.1), provide little help in determining the values of the performance measurement coefficients.

One technique for supplying these values might be direct, physical measurement. The times at which each piece of cargo are picked up and dropped off while enroute could be recorded and translated directly to performance measurement coefficients. This technique might be ideal for situations where the paths over which cargo will travel are known with certainty.

For MAC's problem, direct, physical measurement may not be possible because of the complexity of the channel route system — and the flight schedule is an inherent component of that complexity. The paths over which cargo will travel depend upon the availability of scheduled flights at given locations *at specific points in time* (6). For example, if cargo would *not* have to wait too long on the ground, a direct flight will provide the most timely delivery to the next stop. If, however, the wait *would* be too long, transshipment using indirect means might be quicker.

These time-path relationships in MAC's problem make it one for which an indirect technique for measuring effectiveness might be useful, if the system can be accurately modeled. Pritsker provides the following reasons why problems like MAC's are difficult to model:

The modeling of complex, large-scale systems is often more difficult than the modeling of physical systems for the following reasons:

1. few fundamental laws are available;
2. many procedural elements are involved which are difficult to describe and represent;
3. policy inputs are required which are hard to quantify;
4. random components are significant elements; and
5. human decision making is an integral part of such systems (18:4).

In MAC's case, any one of the five reasons above could apply. Furthermore, these reasons could just as easily be used to describe problems where indirect performance measurement is particularly useful for understanding system operations. According to Pritsker, "simulation models are ideally suited for carrying out the problem-solving approach"(18:5) for these difficult problems because the technique allows "observing the dynamic behavior of a model by moving from state to state"(18:6) as time progresses.

Because of the potential for extracting information about the internal operation of a model, development of a simulation methodology for approaching the MAC scheduling problem could provide useful delay information from a model of MAC's system. Specifically, a simulation methodology might allow determination of *each* amount of enroute delay experienced when a given flight schedule is used. Once performance effectiveness information is obtained about the MAC system, some other technique might use that information for schedule evaluation and for generation or selection of better schedules.

Summary

Many researchers have spent great effort developing techniques to solve problems related to the routing and scheduling of vehicles. This effort continues because of the monetary cost of delivering cargo and the need for efficiency of operations. To solve any specific problem, the research must carefully evaluate the problem and

determine how the methods previously developed can be adapted to the situation at hand — particularly methods for defining system performance.

Because of the schedule variability and the need to re-accomplish the scheduling procedure every month, the performance measurement technique must be dynamic and flexible. Finally, because the MAC timeliness standards apply to individual pieces of cargo, performance measurement has to reflect the manner in which each piece of cargo travels from its place of origin to its destination.

III. Development of the Methodology

This chapter explains the method proposed for gathering information which might highlight where, within a MAC channel cargo schedule, potential exists for improved flight scheduling resulting in less overall cargo delay. The first section describes the assumptions of the method. The second section provides a detailed description of the method. The last section describes the similarities and differences between the methodology of this research and the methodology employed within MAC's current simulation program.

Assumptions

An important aspect for the method development is the relationship to the current MAC computer channel route model. That model is based on certain decision rules and assumptions implicitly affecting the proposed methodology. The computer channel route model also provides information and results upon which the proposed method is based.

The linear programming relaxation applied in MAC's model determines which of the myriad missions making up the entire system should be used for a particular month. Even though changes to a mission schedule might conceivably promulgate adjustment to the mission set selected, neither MAC's simulation nor this research's methodology is intended to support such changes.

This research assumes a monthly schedule (like one provided from HQ MAC/XPYR's scheduling program) forms a starting point from which determination of cargo delay can be based. Using the mission schedule provided by a scheduling program, a more detailed schedule can be built detailing each *flight* necessary for the system. This detailed schedule might then provide a simulation program all the information necessary to "*fly*" all aircraft *flights*.

As with HQ MAC/XPYR's computer channel route model, decision rules establish relationships between types of cargo and specific missions. When an aircraft stops at a cargo unit's point of origin and stops, later in the route, at the cargo's destination, ensuring the cargo gets onboard aircraft following this route makes logical sense. Any model must inherently rely on decision processes like this. Both models attempt to mimic the human decision making process integral to the real system's operation.

The estimated cargo tonnage demanded for the O-D pairs is provided through a computer file. (An example demand file can be found in Appendix P.) Cargo is assumed to enter the system, in one-ton units, evenly throughout the month. All cargo is generic with respect to physical dimension and priority needed for delivery. The order in which cargo is taken out of airports and placed onboard aircraft follows a *First-In, First-Out* (FIFO) rule. Units of cargo flowing through the real network may not be completely described by any current model.

Like the MAC model, this research assumes that loadmaster activities are included as part of the time required to put cargo on or take cargo off aircraft. This research makes no attempt to model the decision logic by which a loadmaster determines to handle certain units of cargo and not others. Without a procedure for directing loadmasters to perform their duties in a particular way, neither the MAC model nor this method can approximate these real decision processes.

The method assumes that the resources used for cargo delivery are always available. Aircraft maintenance allows airplanes to be ready for flying at scheduled departure times. Aircraft fuel can always be obtained. Enough aircrews exist for flying all flights. Any delay associated with ensuring that these resources are available at the proper time is not included.

The monthly (i.e., 720 hour) schedule breaks down into discrete instants of time. Each event (e.g., aircraft takeoff or landing and cargo pickup or dropoff) occurs at one of these instants of time.

Both models assume quantifiable information used as input for the simulation is deterministic, including cargo demand estimates and aircraft flying and ground activity times. This assumption obviously greatly simplifies the actual system in an attempt to provide efficient solution to this highly complex problem.

The decision rules necessary to provide fundamental decisions determining which cargo can travel on which missions can be directly provided by the user. This allows predetermination of "common sense" decisions through quick scanning of the potential route system. The method also allows the computer program to make determinations where no predetermined user decision has been made.

Because of the method used to describe cargo and flights for this research, units of cargo are assumed capable of reaching their destinations in no more than eighteen stops. The current MAC model does not rely on a similar assumption. However, for the actual network, this limitation may not be significant, as routes chosen appear to allow complete cargo travel in less than this number of stops.

In sum, the assumptions underlying this research take on a variety of forms, from philosophical to operational. The applicability of the methodology's development relies on whether these assumptions accurately reflect the important aspects of the real situation.

Development of the Simulation Model

Because this research is intended to gain information which might allow adjustment of specific *flights*, the need existed to use a schedule whose breakdown is by individual flight leg. The FORTRAN programs reproduced in Appendices A, B, and C provide a multiple-month schedule which specifically characterizes each separate flight for a simulation program. The fields of data used to describe each flight are as follows:

- *Field 1* - Mission Number

- *Field 2* - Flight Number
- *Field 3* - Tail Number
- *Field 4* - Aircraft Capacity (tons)
- *Field 5* - Traveling Direction
- *Field 6* - Home Base
- *Field 7* - Stop Number
- *Field 8* - Departure Location (same as home base)
- *Repeating Fields*
  - *9,12,...,45* Ground Time at Stop (first is flight departure time)
  - *10,13,...,46* Flying Time to Next Stop
  - *11,14,...,47* Next Stop Location

By fully utilizing all 47 fields available, each flight defined in the schedule file can fly twelve legs after departing the home base. Appendix Q shows an excerpt of a flight schedule defined in this manner.

The Flow of Events. The events through which the delivery of cargo takes place happen as a result of aircraft departures and arrivals at airports. Cargo does nothing in and of itself - it is acted upon. Each flight follows a general flow of events as shown in Figure 3.1.

A flight departing a location allows cargo to be placed onboard when certain conditions are met. First, space must be available on the plane. The plane must travel in the same general direction the cargo is headed. Essentially, the simulation attempts to mimic the real situation by asking the question: "Are you going my way?" When a flight is traveling in the direction which the cargo needs to go, the cargo will be assigned to the flight. For situations where cargo needs to travel on specific missions, the flight must be one of the required missions.

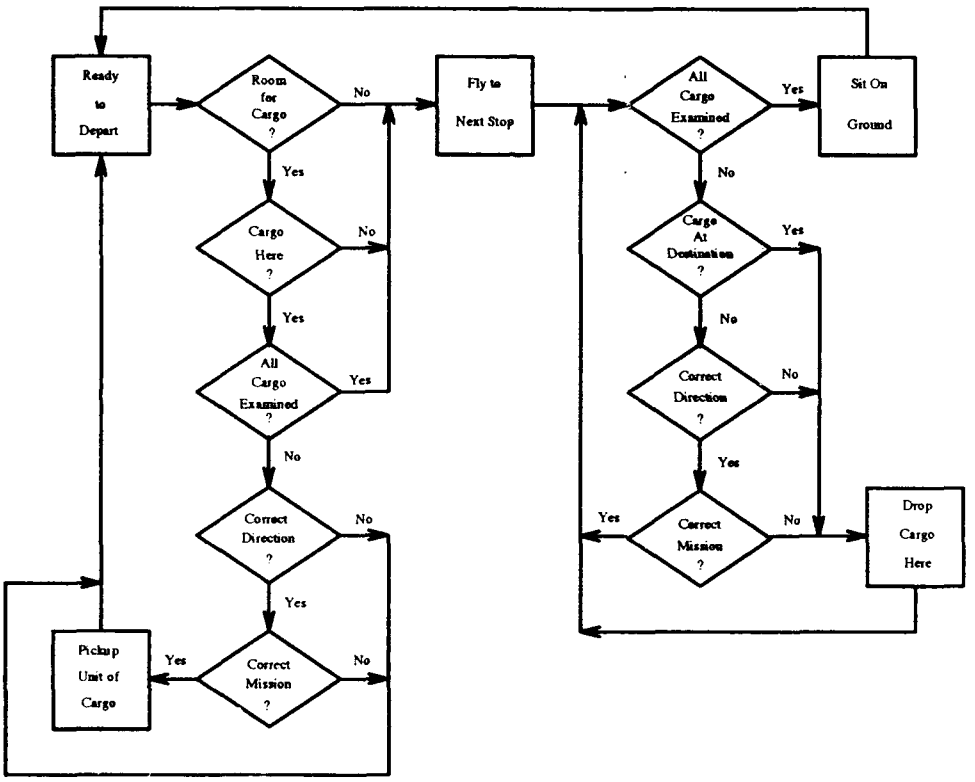


Figure 3.1. Flow of Aircraft Activity

After filling up to capacity or picking up all cargo waiting at the airport which the plane can ship, the aircraft will fly to the next location. When the aircraft stops at the next airport, three different events can occur.

If the cargo has arrived at its final destination, the cargo obviously gets off the aircraft at that location. If the aircraft's mission continues in the direction for which the cargo is destined, the cargo remains on the plane. On the other hand, when arriving at a transshipment location, cargo must determine whether to stay on the aircraft or not. Transshipment decisions play a great part in the delay encountered by cargo enroute to their destinations.

At some point during an aircraft's mission, the direction traveled by the aircraft will reverse — reflecting the return to the home base, for instance. Cargo not intending to return in the direction of origination will get off when the aircraft's direction no longer becomes conducive to delivery. In other instances, cargo may arrive at an intermediate transshipment location where another, more direct, mission will take it toward the destination. In these cases, the cargo will also disembark and await the appropriate mission.

After waiting on the ground for the period of time required for cargo offloading, fuel onloading, aircrew changes, or aircrew rest, an aircraft will look for additional cargo to onload and continue the flight sequence in the manner just described. When the flight completes the route by returning to the home base, all cargo onboard must be offloaded to await future shipment on another flight.

The Simulation Program Operation. A computer program was developed which simulates the operation of a channel cargo route system using "the *s*imulation *l*anguage for *a*lternative *m*odeling, SLAM II" (18:2). The essential flow of events through which the simulation operates is provided in the SLAM code found in Appendix D. Certain aspects of this system could not be modeled directly in the SLAM II language. However, as Pritsker mentions, "Since SLAM II is FORTRAN based, it is a rela-

tively simple matter to add new functions to the language" (18:429). Therefore, the researcher developed a FORTRAN "user-written insert" (18:291) for these purposes.

In SLAM II, "an entity is any object ... which defines or can alter the state of the system" and "can be assigned *attribute* values" (18:98). For this research, the entities for the channel route system are aircraft on specific flights and units of cargo. Most information about the aircraft flights is known before a month begins (e.g., where stops are made, how much cargo can be carried, the time when the flight departs the home base). Similar information about cargo may not be known in advance. For instance, knowledge of the amount of time a particular unit of cargo will wait somewhere before being picked up for the next segment of its journey may not be available. This simulation finds out this type of information.

Attribute fields for aircraft and units of cargo are different. The information included in the attribute fields applicable to aircraft flights has already been shown in the earlier discussion of the flight schedule. A breakdown of the attribute fields for *each* unit of cargo follows:

- *Field 1* - Mission To Get On Flag (0 indicates any mission, -1 indicates specific missions required)
- *Field 2* - Origin Airport
- *Field 3* - Destination Airport
- *Field 4* - Direction Headed
- *Field 5* - Flight Number Currently On
- *Field 6* - Tail Number Currently On
- *Field 7* - Current Location
- *Field 8* - Time Departed Origin
- *Repeating Fields*

- *9,14,...,94* Flight Taken to Next Stop
- *10,15,...,95* Tail Number Taken to Next Stop
- *11,16,...,96* Next Stop Location
- *12,17,...,97* Time Arrive Next Stop
- *13,18,...,98* Time Departed Next Stop

When cargo entities are created, only a few attribute values are assigned. These fields (e.g., 2, 3, 4, and 7) serve as the minimum information necessary to start cargo on their journeys. As each unit of cargo flows through the system, each might "*fly*" on various aircraft and stop at various airports before arriving at the intended destination. The repeating attribute fields are designed to capture information providing a log or history of each unit of cargo's entire journey. The information describing each successive leg is filled in as the journey progresses.

The SLAM Code. The computer instructions needed to operate the simulation are reproduced in Appendix D. Many of these computer instructions appear redundant, because of the need to create cargo for all the different O-D pairs. The remainder of the code, shown below, tells the simulation how aircraft activities are to be performed.

```
FILL EVENT,1,1; update cargo already onboard for next leg
PNEW EVENT,2,1; pickup new cargo for next leg
 FLY ACT/2,USERF(2),; fly aircraft on next leg
     ASSIGN, ATRIB(7)=ATRIB(7)+1.; flight now at next stop
      ACT,0.,1.,DROP;
DROP EVENT,3,1; cargo now told at next location, drop at final
destinations
FDIR EVENT,4,1; change direction of aircraft when required
CDIR EVENT,5,1; change cargo direction and which missions allowed
DOFF EVENT,6,1; drop cargo at transshipment points when required
FFIN EVENT,7,1; drop all cargo when aircraft route complete
     GOON/1;
 SIT ACT/3,USERF(4),USERF(3).NE.ATRIB(6),FILL; sit on ground
LAST ACT/4,0,USERF(3).EQ.ATRIB(6); at last stop
     TERM;
     .
```

```
      .
      .
DONE QUEUE(21),0,,,;
     ACT(1),0.,1.,;
CARG EVENT,8,1; information gathering for output
     TERM;
```

This computer code implements the activities shown in Figure 3.2. The solid lines in Figure 3.2 represent the flow of activities happening to the flight and cargo entities directly. For example, the simulation will "*fly*" the planes or "*sit*" them on the ground. The dashed lines indicate logical relationships allowing actions carried out indirectly through specialized FORTRAN code. A more detailed explanation of the FORTRAN code is provided in the next section. What can be seen from Figure 3.2, however, is that much of the operation of the model does not occur from the simulation code directly, except for the aircraft flight movements from airport to airport.

The boxes with the diagonal lines represent *QUEUEs*, or places where the units of cargo are waiting. The simulation establishes a computer file for each *QUEUE*, where an "entity's attributes and the relative position of the entity with respect to other entities waiting is maintained" (18:116). The *QUEUEs* in this simulation are only used for storing units of cargo. "FIFO is the default priority for files" (18:116) and is used for all *QUEUEs* here. Each airport in the system is represented by a separate *QUEUE*.

Two of the other three *QUEUEs* used in the simulation are shown in Figure 3.2. These are the *DONE* and *HOLD QUEUEs*. All cargo arriving at their destinations are processed through the *DONE QUEUE* for information gathering. The *HOLD QUEUE* maintains *all* cargo onboard *all* flights at a particular point of time. For further discussion of these *QUEUEs*, see Appendix F.

Most of the simulation's activity puts cargo into particular *QUEUEs* and takes cargo out of *QUEUEs* as aircraft "*fly*" through the network. When an aircraft flight entity passes through each box of Figure 3.2, such as the PNEW box, a number of

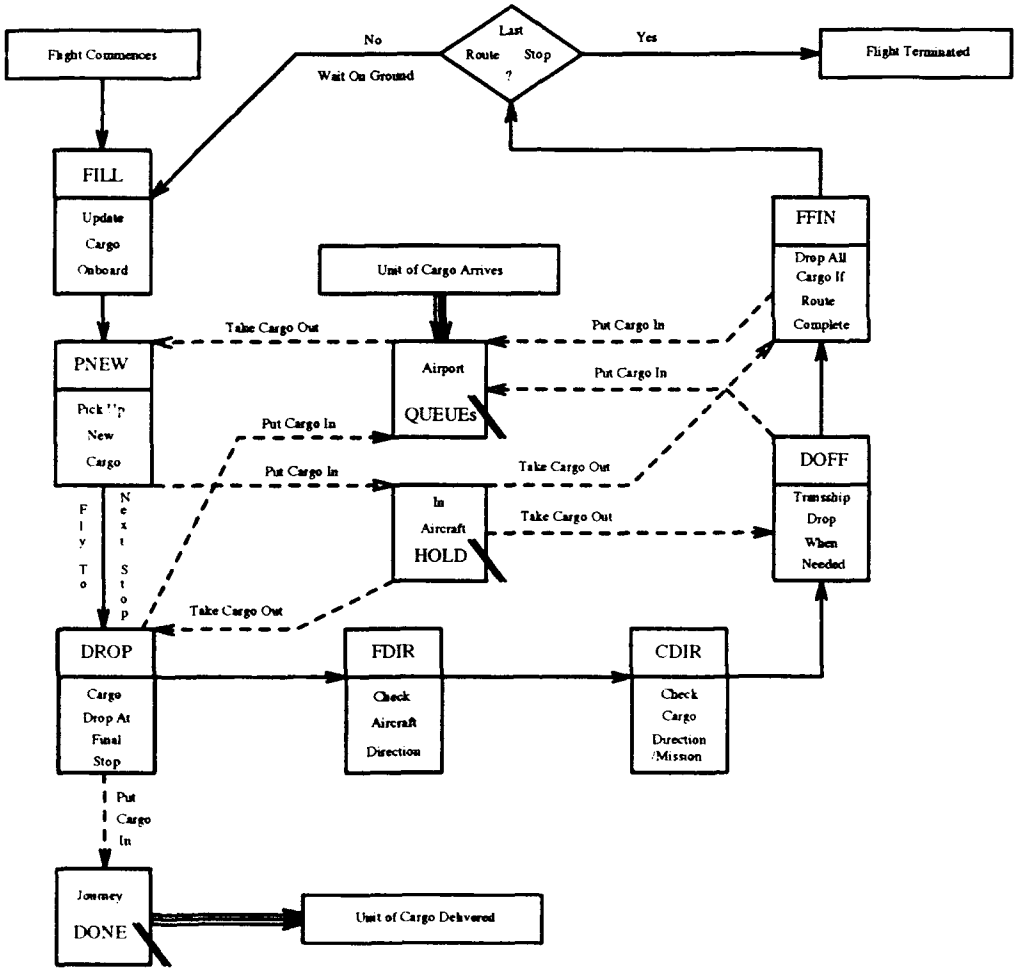


Figure 3.2. SLAM Logical Flow

computer processes occur at the FORTRAN insert level of the simulation to carry out these cargo placements.

The FORTRAN inserts. The FORTRAN code for the simulation (reproduced in Appendix E) provides for a number of functions which the particular simulation language does not directly provide. The reason for many of these FORTRAN insert functions revolves around SLAM's inability to directly access and change the attribute values of two different network entities at the same time. Although the simulation language provides a way to measure average cargo delay, providing non-averaged cargo delay information required the development of FORTRAN subroutines specifically designed for this purpose.

The FORTRAN code makes all the decisions shown in Figure 3.1 as the flights follow the sequence of events depicted in Figure 3.2. More importantly, as decisions are made to assign cargo to specific flights, the FORTRAN code inserts information into the unfilled repeating attribute fields for the units of cargo. The information placed in these repeating fields is provided by the attributes of the flights used and the simulated time. One might think of the code as transferring flight attribute information to cargo attribute information as the units of cargo travel through the system.

As an example, consider the activities transpiring when a flight looks to pickup new cargo at an airport (i.e., the *PNEW* block in Figure 3.2). The flight's attribute values define the airport at the flight is currently located. Once the location is determined, the FORTRAN code peruses the cargo waiting at that airport, looking for cargo which needs to travel in the correct direction and on the mission type to which the flight is assigned (i.e., as depicted on the left side of Figure 3.1). When cargo is found meeting these criteria, the FORTRAN code takes the cargo from the current airport; assigns values to the cargo's next, unfilled set of repeating attribute

fields (e.g., the flight taken, the time picked up, etc); and puts the cargo in the cargo *HOLD*.

Each activity block in Figure 3.2 (e.g., *PNEW, DOFF*) corresponds to an entire subroutine in the FORTRAN code, providing the computer instructions necessary to process the decisions which must be made as each flight flies along its route. Two other subroutines provide the cargo delay and flight leg utilization information.

The *DONE* queue block shown in Figure 3.2 represents units of cargo reaching their destinations. Once there, the FORTRAN code essentially breaks down the journey by each set of repeating attribute fields. Using this information delay can be measured from the time each unit of cargo departed the previous stop until departing the present stop. The delay-reporting subroutine then groups similar cargo (i.e., same O-D pair, put onboard the same plane, at the same airport, at the same time). Enroute delay is calculated for each group, subtotaled for each O-D pair at each airport, subtotaled for each O-D cargo pair, and totaled for all cargo.

Having specific information about cargo delay does not provide information which might lead to flight changes resulting in better schedules. For this reason another subroutine looks at the cargo carried onboard each flight as each route leg is flown. This subroutine reports grouped types of cargo carried (by O-D pair and time put onboard) for each flight leg.

As can easily be seen by the dotted lines in Figure 3.2, the great majority of logical activity occurring within the simulation occurs as a result of the pickup and dropoff of cargo. All of this activity occurs within the FORTRAN portion of the simulation code. For a more detailed explanation of the computer instructions used to carry out these activities, see Appendix F.

Comparison to the MAC Simulation

The simulation used in MAC's current channel cargo model (hereafter referred to as MAC's simulation) and this research's simulation perform the same general

function — computer simulation of cargo pickup and dropoff by aircraft flying within a route network. (The simulations use different simulation languages, although either language might be used.) These simulations model some aspects of the problem similarly, while other aspects are treated differently. The significant entities modeled in both simulations are aircraft and units of cargo.

Both simulations characterize the entities which can hold cargo (e.g., aircraft) as having certain attributes which may be different from one entity to another (e.g., home base, capacity, etc.). In MAC's simulation, the important aircraft entity is a tail number, while this research focuses on each flight. The one-ton units of cargo, used in both simulations, attempt to travel from their origin points to their destination points along the most direct paths through the system using decision rules provided by the simulation user.

The manner in which the two simulations depict the entity characteristics varies considerably. Both methods rely on certain attributes which never change (e.g., aircraft capacity and cargo O-D pair information) and some attributes whose values change as the entities move throughout the system. The difference in attributes relates to the number of attributes required to describe an entity and to the necessity for changing the attribute values at different points in the simulation.

The entity attributes used in MAC's simulation focus forward in time. No information is maintained with either aircraft or cargo to describe the path used to arrive at a particular location. Instead, the attributes supply information used for determining how travel might be accomplished from the current location in the future. In particular, as a unit of cargo travels through the system, minimum information must be maintained to adequately describe that unit of cargo with respect to where it must go from the current location.

This research's simulation, on the other hand, relies upon a much larger number of attributes. These attributes supply not only the characteristics applicable to the entity's current location and orientation within the system, but also the information

describing the entity's entire travel history. In this research's simulation one specific entity can be fully described only by all information describing its entire path through the system.

The simulations rely on user-provided criteria through which decisions assign cargo to aircraft for shipment. Both simulations rely upon a two-level decision-making framework. As previously discussed, this research's framework involves the cargo's direction of travel and available missions traveling toward the destination airport. MAC's simulation performs essentially the same operation in a slightly different manner.

For this research's simulation, the user must determine (ahead of time) which direction cargo must travel from one location to arrive at another location, and whether assignment to particular missions will allow more direct shipment. For the MAC simulation the user must determine (ahead of time) which airport must be visited after the current airport, and whether assignment to particular missions will accomplish this goal. Thus, the difference in the two methods revolves around determination of *direction of travel* as opposed to *place to travel to*.

Both methods used to simulate channel cargo route system operations simplify the real-world situation. Neither simulation addresses the issue of the stochastic nature of activity times. (This research's simulation also does not differentiate the cruising speeds for different types of aircraft, which impacts flying time).

Furthermore, both simulations carry out channel operations as if a month's cargo demand and flight schedule continue indefinitely into the future. These two simplifications directly affect the amounts of cargo in the system as the simulations operate and the amounts of time taken for cargo to travel to their destinations.

This research's simulation does not report cargo delay in the same manner as the MAC simulation. Because the MAC simulation does not keep track of the times associated with cargo arrivals and departures at *every* airport where cargo stops, no

information remains available to calculate the amount of delay experienced *within* the journey from the origin to the destination. By maintaining the minimum number of attributes to distinguish pieces of cargo, MAC's simulation can normally calculate only a moving average delay for each O-D pair as cargo arrive at their destinations. Using the much larger number of attributes, this research's simulation reports *every* instance of delay by *all* cargo, cross-referenced to the time at which units of cargo experienced delay, to specific location, and to the flight departing at that time. The time, location, and flight cross-reference provides information for determining where adjustment of the monthly flight schedule might result in improvement — information not available from MAC's simulation.

Summary

This chapter has provided a description of the logic used to develop the computer programming instructions necessary to simulate the operation of a channel cargo route system. Assumptions were presented which focuses the simulation's results on obtaining information specific to the purpose of this research. The FORTRAN programs and SLAM simulation model developed for this research depict the activities transpiring as aircraft flights pick up and drop off units of cargo.

Neither MAC's simulation nor this research's simulation accounts for all factors affecting the operation of MAC's channel cargo route system. Both, however, simulate network operations by allowing a method for application of user-supplied decision rules for putting enroute cargo onboard aircraft to provide delivery service to the intended destinations. These two simulations are similar, except for what decision criteria are required and what information is maintained to describe aircraft and units of cargo as they flow through a channel route network. This difference in information maintained, particularly about cargo, forms the reason why the two simulations report cargo delay differently.

IV. Methodology Testing And Analysis

To show the results obtainable, this research developed a hypothetical channel cargo route system. This chapter provides the information describing this twelve-airport network and the results obtained when using this research's information-gathering methodology on the operations of the hypothetical system.

The Twelve-Airport Network

Based on MAC's channel cargo route system, a hypothetical twelve-airport network was developed with cargo demands occurring from a variety of O-D pairs and with aircraft flying a number of different missions to service the cargo demands.

By analyzing historical information provided by HQ MAC/XPYR, patterns of cargo demand were found which impacted the choice of cargo tonnage amounts included in the hypothetical system. Demand over certain O-D pairs is much greater than (e.g., thousands of times larger) demand over other O-D pairs. Because of these order-of-magnitude variations, cargo demand for certain O-D pairs supply the bulk of the cargo in the entire system. The amounts of cargo demanded in the hypothetical system are shown in Table 4.1. The computer file which the simulation accessed for demand information is reproduced in Appendix P.

The amounts of demand shown in Table 4.1 are intended to be representative of the demand experienced in a real channel cargo system. In this small system, most of the cargo demand is between airports *1* and *6*. A variety of demand levels exist between other airports. In fact, no demand was established for airport *11*, which represents a transshipment point only. And as can be seen in Table 4.1, some O-D pairs have so little demand as to seem almost insignificant. Demand for these O-D pairs, however, may prove to be significant when the units of cargo must compete for aircraft space with the much higher demand for other O-D pairs.

Origin	Destination											
	1	2	3	4	5	6	7	8	9	10	11	12
1				73		2067			183			250
2			300		112		130	223		40		122
3		209		4		43				6		
4	46					9			9	6		
5		72		10			18	32				23
6	1161		62	27					183	11		
7		47			28			32				
8		145		9	50		35					
9	61			12		21						
10	29		8	9		4						
11												
12	240	52			15							

Blanks indicate no cargo demand between O-D pair.

Table 4.1. Tons of Cargo Demand by O-D Pair

Figure 4.1 depicts the entire twelve-airport system. The larger circles represent the airports. The lines represent the paths over which aircraft fly, and each route leg is shown with the mission number to which it is assigned. All fourteen missions used in the small route system are shown in Figure 4.1. The different types of lines provide differentiation between aircraft types: solid lines represent C-5 aircraft, dashed lines represent C-141 aircraft, and dotted lines represent C-130 aircraft.

The distances between airport locations for which cargo demands occur cover the entire range of possible separations within the network — some O-D pairs are located right next to each other, some O-D pairs are at opposite ends of the system, and others are somewhere in between.

Aircraft home bases were chosen at only a few of the airports of the system since monetary and political costs associated with housing maintenance facilities and aircrew personnel at a multitude of airports could be prohibitive. In this smaller system, C-5 and C-130 aircraft are based at airports *1* and *9*, respectively. C-141

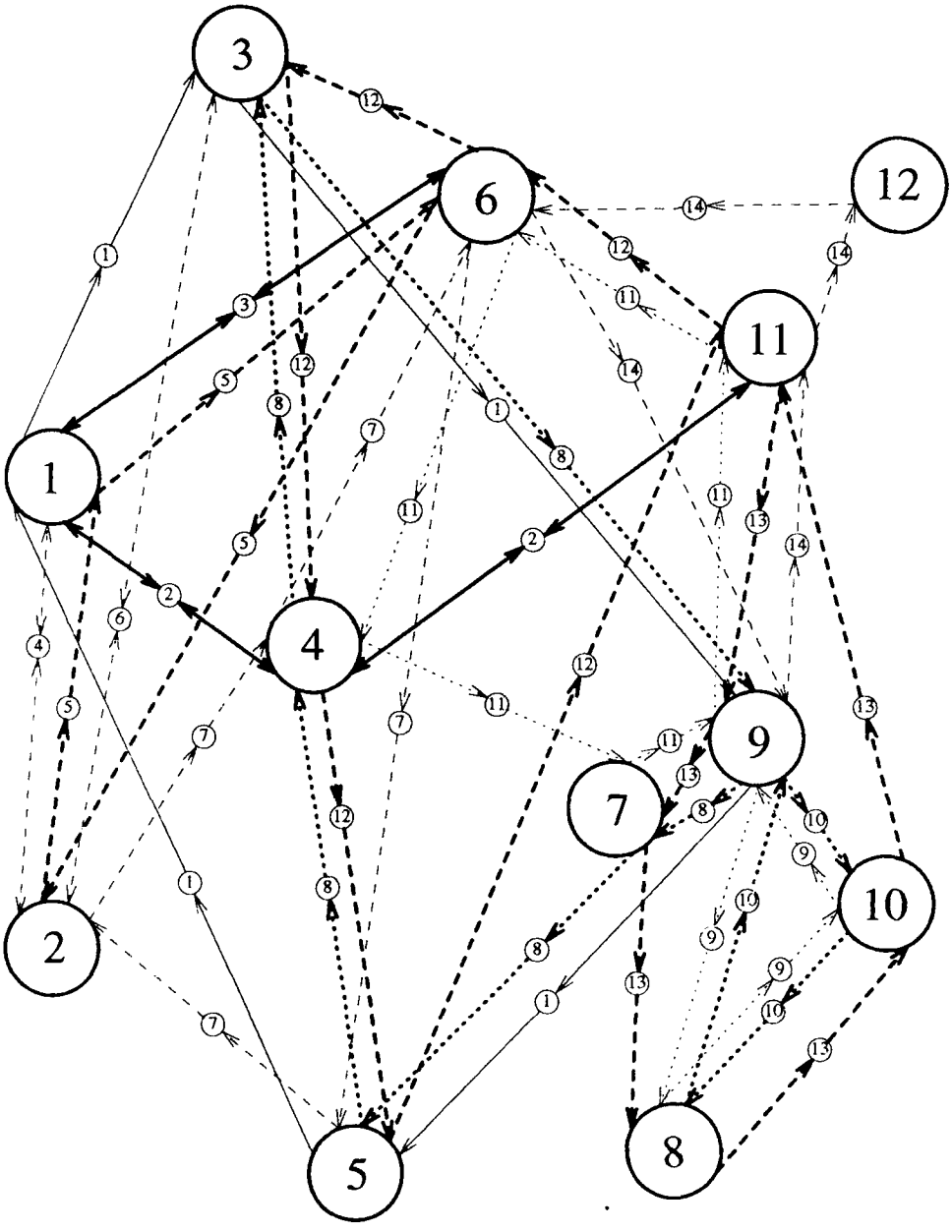


Figure 4.1. The 12-Airport Channel Route System

aircraft are based at two locations (i.e., airport *2*, and airport *11* where transshipment facilities are available).

The routes in the hypothetical system were designed to provide a system exhibiting certain characteristics. Because of the need to deliver enormous quantities of cargo to certain locations, multiple routes might follow the same path, although not necessarily in the same direction. Certain routes follow an out-and-back path while other route paths are more circuitous. Some routes allow for direct delivery of heavy-demand cargo. Some routes selected for large capacity (e.g., C-5) aircraft provide long-haul shipment between the major airports of the system. These *trunk* routes connect to local *feeder* routes, supplying delivery service to airports close to the *trunk* route stops. In fact, certain route selections force cargo to travel on specific missions to depart and arrive at some airports. The routes chosen for this twelve-airport system are provided in Appendix H and shown in Figure 4.1.

Additionally, the direction of travel along routes was specifically tailored to provide as much direct shipment as possible and to force cargo transshipment to occur at a variety of locations throughout the system. For example, in this system the majority of cargo demand occurs between airports *1* and *6*. For this O-D pair, missions 3 and 5 allow for direct shipment. However, any cargo at airport *2* enroute to airport *5* must transship through another airport before arriving at airport *5*.

Along with route selection, particular types of aircraft were assigned to each route. In general, the larger (e.g., C-5) aircraft were assigned to the longer routes, although this is not necessarily so. Smaller (e.g., C-130) aircraft may have longer trips, but usually require extra stops along the route for purposes of crew rest (the reason for stopping which requires the longest ground time).

The jumble of lines interconnecting the airports in Figure 4.1 might appear confusing. Any perceived confusion is inherently implied as a result of the apparently overlapping routes used within the system. Figure 4.1 simply shows a small example

of one of these kind of networks. MAC's route system, with over 100 airports, seems just as confusing.

The number of flights for each mission in the hypothetical network were calculated to approximate the number of missions which would have been selected by the linear programming relaxation of MAC's computer channel route model. Particular emphasis was placed on delivering the O-D pair demand which formed the bulk of the total system demand. The number of flights for each mission can be found in Appendix I. The specific number of mission flights required by this system might appear large, but this came about simply from the amount of cargo demand hypothesized. MAC's system may not have as many of any one type of mission, but it certainly generates more total flight requirements.

Of particular concern in the development of this hypothetical network was the need to incorporate as many aspects of MAC's system as possible. As such, the hypothetical system operations may be somewhat dense. In other words, almost anything that can happen does happen — almost everywhere. In the larger MAC system, some subsets of the system may operate similarly to this hypothetical case. In other subsets, very little activity might occur. This hypothetical system thus may not mimic systems where there is sparse activity.

This small hypothetical route system exhibits many of the aspects of MAC's channel cargo system including: different types of aircraft flying on a variety of missions; wide disparity in the amount of cargo tonnage demanded between O-D pairs; routes, selected for a variety of reasons, providing for direct shipment and transshipment of cargo; and most importantly, a system through which cargo assignment decision rules allow for cargo to travel along a variety of paths to reach their intended destinations.

The complexity of the MAC computer channel route model can be approximated using a much smaller route network (i.e., one encompassing only twelve airports). This complexity involves the types of routes flown, the physical relationships

between the routes and the O-D pair shipments, and the shear number of flights required. The MAC model actually generates the flights which are flown in the real system, whereas this research can only approximate those results.

Results From The Twelve-Airport System

The data describing the hypothetical cargo network (contained in Appendices G through P) were processed by the simulation model developed for this research. As anticipated, cargo delay was experienced throughout the system during the month of data collection (i.e., month two of a three-month schedule). The subtotal and total amounts of cargo delay calculated for the system are depicted in Table 4.2.

	Destination											
Origin	1	2	3	4	5	6	7	8	9	10	11	12
1				515		15302			8406			44150
2			4544		7270		13263	22274		4106		21354
3		2027		6		4005						
4	400					21			519	539		
5		2247		652			639	1147				4628
6	17303		86	4424					1275	298		
7		4004			283			45				
8		14460		1607	3640		1652					
9	1677			940		238						
10	1644		888	1083		197						
11												
12	7176	1072			644							
Total Hours Delay (All Cargo)												222647

Blanks indicate no cargo demand between O-D pair.

Table 4.2. Hours of Cargo Delay by O-D Pair

As can seen in Table 4.2, the amount of delay applicable to different O-D pair units of cargo varies considerably. The figures in Table 4.2 are weighted by O-D pair, because each unit of cargo carries the same weight (i.e., one ton) regardless of which O-D pair it belongs to. Direct comparison of these delay amounts might be misinterpreted if viewed without reference to the number of units flowing between each pair. (See Table 4.1 for each O-D pair tonnage.)

Some of the delay experienced in the twelve-airport system highlight the possibility that further investigation of simulated operations may be in order. For instance, although the cargo tonnage shipped each way between airports *1* and *12* are nearly the same, Table 4.2 shows the amounts of delay associated with the two O-D pairs to be vastly different. Another interesting result is the cargo traveling from airport *3* to airport *10* – none of it arrived during the entire month of operations. (Although note must be made that this cargo accounts for only one-tenth of one percent of the cargo flowing through the system.) This anomalous result may indicate the need for further investigation into the way simulated cargo flows through the channel system.

For each subtotal figure in Table 4.2, the simulation provided information about the breakdown showing where exactly the delay occurred. Tables 4.3 and 4.4 show examples of this type of breakdown.

	Destination			
Delay At	4	6	9	12
1	514.80	15302.33	4833.55	4339.30
3			886.60	7016.74
4			2580.90	8163.20
7				131.25
8				29.05
9				20993.16
10				2962.30
11			105.00	514.60
Totals	514.80	15302.33	8406.05	44149.61

Blanks indicates cargo did not stop at location.

Table 4.3. Delay Hours Experienced by Cargo Originating at Airport 1

As can be seen from Table 4.3, delay for some O-D pairs occurs at only one airport while delay for other O-D pairs occurs at a number of airports. The fact that cargo traveling from airport *1* to airport *6* only experiences delay associated with

airport *1* is the direct result of the mission decision criteria used for the hypothetical route system. (As found in Appendix O, this type of cargo was forced to travel only on direct missions.) In the case of cargo traveling from airport *1* to airport *12*, multiple paths exist by which cargo can (and did) make this journey. Thus, the last column of Table 4.3 shows a number of airports where cargo was delayed and by how many hours.

Table 4.4 shows another interesting example of the breakdown by location where delay occurred at various airports. In the case of cargo originating from airport *8*, no matter where cargo is destined, there are multiple paths over which cargo traveled.

	Destination			
Delay At	2	4	5	7
1		31.20		
3	892.40		527.80	
4	274.50		23.80	
5	3054.30	621.00		
6	2561.85		857.50	
7		18.55	4.60	
8	1247.50	88.25	337.10	468.95
9	1137.70	91.75	56.00	59.50
10	3158.50	739.50	1075.50	1014.10
11	2077.00	17.20	757.30	109.25
Totals	14459.85	1607.45	3639.60	1651.80

Blanks indicates cargo did not stop at location.

Table 4.4. Delay Hours Experienced by Cargo Originating at Airport 8

Information like that shown in Tables 4.3 and 4.4 provides the data required to determine the locations where delay exists when the simulated network operates. In order to make a better schedule for the route network, focus might then be directed to decrease delay resulting at specific locations. This delay might result because of a buildup of cargo awaiting transportation. Or, the delay might simply result from

the non-availability of flights stopping at the location. In any case, the breakdowns shown in Tables 4.3 and 4.4 do not provide enough information to determine if the delay is significant, to assess whether schedule adjustments will reduce the amounts of delay, or to make schedule adjustments to individual flights.

To decide how modification of a flight schedule might affect the simulation operations requires more detailed information. For this reason, the simulation provides more detailed information than the results shown in Tables 4.3 and 4.4.

Tables 4.5 and 4.6 show this type of information. Before discussing some of the implications of the data in these tables, a reminder must be made about how the information was obtained. As discussed in Chapter III, the use of cargo attribute values allows reconstruction of the flight paths of *each* unit of cargo completing the journey from origin to destination. The information found in Tables 4.5 and 4.6 (and the excerpt of one of the simulation's output files found in Appendix R) is based *only* upon units of cargo which have *completed* their journeys and have been processed through the last simulation event. This should explain the references to flights departing before the beginning of the second month.

Table 4.5 shows one of many situations which occurred within the twelve-airport route system. For those 34 tons of cargo, which were able to fly from airport *4* to airport *5* on a direct flight, the only delay experienced was because of the flying time taken to make the trip. This should not imply that all the cargo going from airport *8* to airport *5* traveled through airport *4*. But for those that did, they experienced as little delay as possible on that portion of their journey. For situations like that shown in Table 4.5, the information may not support the need for schedule adjustment to the associated flights.

On the other hand, Table 4.6 shows somewhat different results. The same type of cargo is the basis for Table 4.6 as for Table 4.5 – cargo going from airport *8* to airport *5*.

Tail Number	Flight Number	Depart Time	Cargo Units	Delay Hours Each Unit	Total Hours This Flight
15	183	733.55	3	0.70	2.10
15	202	793.55	3	0.70	2.10
15	212	853.55	3	0.70	2.10
15	229	913.55	2	0.70	1.40
15	246	973.55	3	0.70	2.10
15	262	1033.55	1	0.70	1.40
15	278	1093.55	3	0.70	2.10
15	296	1153.55	3	0.70	2.10
15	307	1213.55	3	0.70	2.10
15	323	1273.55	4	0.70	2.80
15	341	1333.55	3	0.70	2.10
15	357	1393.55	3	0.70	2.10
Subtotal Delay					23.80

Transshipment At Airport 4

Table 4.5. Delay Hours Experienced by Cargo O-D Pair 8-5

In the instance depicted in Table 4.6, some units of cargo apparently traveled on direct flights (as indicated by the 4.75 hours of delay per unit) while other units of cargo did not. In particular, the two units of cargo which were picked up by flight 256 experienced a very large amount of delay in comparison to all the other units shown. The obvious explanation for this delay is that these units of cargo apparently arrived at the airport after the other units of cargo which were picked up before them. Since the assignment of cargo to aircraft is assumed to follow a FIFO rule, these units could not depart the location until all other units of cargo which could travel onboard similar missions were exhausted.

Results similar to that contained in Table 4.6 might provide the information necessary to make schedule adjustments. As an example, a potential adjustment might be to push the leg which flight 256 flies out of airport 6 forward in time. Another possibility might be to move all flights prior to flight 256 forward in time, to clear out all cargo waiting. To make any of these adjustments, however, the

Tail Number	Flight Number	Depart Time	Cargo Units	Delay Hours Each Unit	Total Hours This Flight
15	183	709.95	1	4.75	4.75
14	181	698.80	2	34.75	69.50
15	202	769.95	3	4.75	14.25
15	212	829.95	3	4.75	14.25
15	229	889.95	2	4.75	9.50
15	246	949.95	3	4.75	14.25
15	262	1009.95	1	4.75	4.75
15	278	1069.95	2	4.75	9.50
15	276	1058.80	1	34.75	34.75
15	296	1129.95	3	4.75	14.25
15	307	1189.95	3	4.75	14.25
15	323	1249.95	2	4.75	9.50
15	256	986.80	2	286.75	573.50
15	341	1309.95	3	4.75	14.25
15	357	1369.95	2	4.75	9.50
14	351	1346.80	1	46.75	46.75
Subtotal Delay					857.50

Transshipment At Airport 6

Table 4.6. Delay Hours Experienced by Cargo O-D Pair 8-5

rules to follow for making schedule changes might need to evaluate the effects on the amount of cargo space utilized on aircraft. The adjustments to any flight legs would, most likely, be made to the original one-month flight schedule, not the three-month schedule. (See Appendices A, B, and C for explanation of these different schedules.) In the case just mentioned, flight 256 flying in the second month corresponds to flight 61 which began at the same relative time during the first month. Thus, an adjusted flight 61 might form the basis for a modified flight schedule.

Table 4.7 displays the amount of cargo space used on one of the 190 flights flown during the second month of simulated operations. Table 4.7 also shows exactly which type of cargo was onboard the aircraft during the legs of the flight.

As mentioned in the discussion of Tables 4.5 and 4.6, potential schedule modification might consider the impact on aircraft utilization. For example, if flight 212's schedule was under consideration, the information in Table 4.7 might be useful. If no other flight times were to change, a potential schedule change might involve delaying the first flight leg in order to use more of the aircraft's cargo capacity. Or, other flights might be delayed to force more cargo onto flight 212's first leg. With the information such as Table 4.7 available for every flight, the process for schedule adjustment might estimate what impact a change in one flight's schedule will have on other flights. Appendix S provides a small excerpt of the simulation's report of what cargo was onboard each flight leg.

The route structure and number of flights used for the hypothetical twelve-airport route system were designed to approximate the output which would result from use of the mission set provided by MAC's linear programming relaxation. Although not the purpose of this research, the results shown in Table 4.7 might provide a type of check on a schedule's ability to utilize the aircraft space available from the mission set. One check for a better flight schedule might be to ensure some percentage of flight legs are nearly fully loaded with cargo. Caution may be necessary with this criteria, however, as completely full flights may just indicate a buildup of

Leg Number	Origin	Destination Pair	Cargo Units	Time Got On
1	8	5	3	807.50
	8	2	6	807.50
	10	1	2	807.50
	9 Tons of 20-Ton Capacity Unused			
2	8	5	3	807.50
	8	2	6	807.50
	2	5	3	829.95
	6	3	5	829.95
	12	5	1	829.95
	2 Tons of 20-Ton Capacity Unused			
3	8	5	3	807.50
	2	5	3	829.95
	12	5	1	829.95
	3	6	2	834.70
	2	8	4	834.70
	2	7	3	834.70
	2	12	2	834.70
	2	10	2	834.70
	Entire 20 Ton-Capacity Used			
4	8	5	3	807.50
	2	5	3	829.95
	12	5	1	829.95
	2	8	4	834.70
	2	7	3	834.70
	2	12	2	834.70
	2	10	2	834.70
	2	5	2	853.55
	Entire 20 Ton-Capacity Used			
5	2	8	4	834.70
	2	7	3	834.70
	2	12	2	834.70
	2	10	2	834.70
	5	12	2	857.50
	5	8	3	857.50
	5	7	1	857.50
	3 Tons of 20 Ton-Capacity Unused			

Tail Number 15 Flying Mission Number 12

Table 4.7. Cargo Onboard Flight 212 by Flight Leg

cargo throughout the entire network, although the linear programming relaxation procedure should account for all anticipated cargo demand.

Summary

A small, hypothetical version of the MAC channel cargo route system developed for this research was explained, and shown to exhibit many of the characteristics of the larger network system. When used with the information describing the hypothetical network, the simulation generated information expected to be useful for analysis of the current schedule. The simulation results provided detailed explanation of which cargo experienced the enroute delay by O-D pair. The simulation also provided information which helps identify the specific airport locations where the cargo waited for aircraft flights and for which specific flights they were waiting. This type of information might then be used to determine how a flight schedule might be adjusted for improved performance.

V. *Conclusions and Recommendations*

This chapter provides a summary of the research and presents ideas for future research involving the use of detailed delay information to improve the schedules in the MAC computer cargo route model.

Conclusions

This research has developed and tested a method for obtaining the detailed information necessary to identify enroute cargo delay associated with aircraft departures within a monthly channel cargo route system schedule. The simulation developed for this research provides the output to recreate *every* segment of *every* unit of cargo's travel history and identify *all* cargo traveling on *all* flight legs as the route system operations are carried out.

The computer instructions for the simulation model perform a large amount of not only file manipulations but also data searches and comparisons. These computer operations might require increased computer resources for a larger channel cargo route system. For the twelve-airport system, the simulation took a VAX 6420 computer 5.83 CPU minutes to report results. A correspondingly more powerful computer system may be required to provide the delay information for a larger network.

By utilizing information like that provided by this research, determination can be made of where potential exists for flight adjustments resulting in enroute cargo delay reduction. By analyzing the flight departures associated with greatest amount of delay, changes to those departures might lead to the creation of better schedules.

Recommendations

Future research might portray a more realistic model of MAC's actual system. For instance, flight times, ground times, and cargo arrival times might be treated

stochastically. Research could focus on the manner in which cargo is described within the system (e.g., how small should a depicted unit of cargo be, how to assign cargo to aircraft where dimensional size is more important than tonnage, how to describe the prioritization of cargo prior to shipping and changes thereto while enroute). Another idea might be to adjust the methodology to account for changes in cargo demand and flight schedules which occur from month to month. Therefore, any research which decreases the assumptions necessary to model the system might allow more realistic appraisal of the cargo delay.

Future research might focus on how the decision criteria used for assigning cargo to aircraft flights affects the ability of a schedule to decrease the enroute cargo delay. Regardless of what type of decision criteria are embedded in the model, the detail with which those criteria are modeled might have significant implications on the way cargo flows in the network, waits for shipment at airports, and therefore, experiences delay. Furthermore, comparing the decision criteria used in the model against those used in the actual system would ensure that the model provides a valid portrayal of actual operations.

As this research measured delay based on units of cargo which had completed their journeys, no information has been obtained relating to incomplete journeys. Future research might attempt to determine enroute cargo delay for these partial cargo journeys.

The focus of this research has been to gain detailed information about enroute cargo delay. The next step in HQ MAC/XPYR's attempt to produce better schedules might be to develop a technique (e.g., a heuristic procedure) for using this information. To decide whether an existing schedule can be improved upon, analysis might focus on aspects of the schedule where modification would impact system performance. After determining the necessary adjustments, the flight schedule could be modified and rerun through the simulation to assess delay associated with each schedule iteration.

This type of iterative procedure could be continued until some degree of schedule goodness (e.g., a maximum acceptable level of total system delay) is attained for the mission set.

Any schedule adjustment procedure which might be developed must address the issue of optimality with respect to the number of aircraft needed to fly all the flights. If adjustments are made to a schedule based on a set number of aircraft, the potential exists to make changes which might require more aircraft to fly all the flights. If this happens, then questions will have to revert to which type of system performance measurement has priority — one based on maximization of usage, or minimization of (monetary) cost; or one based on the maximization of timeliness, or minimization of delay.

Eventually research might look for a method to measure schedule timeliness as the monthly mission structure is determined. The choice of routes may implicitly impact a schedule's timeliness, and vice versa. Therefore, as the mission set to be used for the month is chosen, some measure of timeliness might help determine whether one particular set is better than other sets.

Appendix A. *The CARGRT.FOR Program*

HQ MAC/XPYR has created this FORTRAN scheduling program to create a schedule by which aircraft can fly their missions. This program uses various data (e.g., aircraft flying times, aircraft cargo capacities, and descriptions of routes used), stored in computer files, to create a mission schedule shell.

Examples and descriptions of the input files used for this program can be found in Appendices G through K. The mission shell for all planes created by the MAC's FORTRAN program is output into computer-file form which can be directly input into their simulation. An example and detailed description of this output file is provided in Appendix L.

This program for mission scheduling determines the minimum number of *planes* which can fly the number of flights called for by the linear programming relaxation. For each plane, the program creates a shell by which the plane will fly each flight in consecutive sequence. This research uses MAC's program intact, with the exception of determination of each plane's initial flight departure time. In the original program, the first plane begins its first flight at the very beginning of the month, while each subsequent plane's first takeoff is delayed 36 hours from the previous plane's first takeoff. For this research the delay between first departures has been cut to 2.5 hours, allowing the program to schedule the appropriate number of flights called for by the linear programming output.

```
c  This is the SCHEDULE BUILDING FORTRAN program!

c*****************************************************************
***
c**      This program builds the route.dat and jet.dat files
**
c**  necessary for running the channel cargo simulation model
**
c**  located on the SUN workstation.  The program takes 5 files,
**
c**  a standard XPYR formatted route file called route.inp, a
```

```
**
c**  second file with the corresponding route frequencies called
**
c**  freq.inp, a base file called base.inp which is a standard
**
c**  location key, a groundtime file called gndtm.inp and a
**
c**  flying time file called fly.dat
**
c**  This program was written by HQ MAC/XPYR personnel.
**
c*****************************************************************
***

      integer maxrts,maxstops,maxbases
      parameter (maxrts=500,maxstops=20,maxbases=200)

      integer nicao(maxrts,maxstops),reason(maxrts,maxstops)
      integer num,numroutes,k,i,j,freq(maxrts),actype(maxrts)
      integer iroute(maxrts,maxstops,5), numstop(maxrts)
      real gtm(7,9),flytm(maxbases,maxbases),route(maxrts,maxstops)
      real origstart
      integer a,b,totac,totlines,multac,ace
      character*4 icao(maxbases)
      character*4 name(maxbases)
      character*4 micao(maxrts,maxstops)

c     ********* Stops are input as ICAOs.  Reasons for ***********
c     ********* stops are as follows:

c     ********* 1) originate (start mission)
c     ********* 2) onload
c     ********* 3) offload
c     ********* 4) enroute fuel
c     ********* 5) enroute crew change
c     ********* 6) enroute crew rest
c     ********* 7) spare
c     ********* 8) spare
c     ********* 9) terminate (stop mission)

      open(unit=8,file='base.inp',status='old')
      open(unit=9,file='freq.inp',status='old')
      open(unit=10,file='route.inp',status='old')
      open(unit=11,file='route.dat',status='unknown')
      open(unit=12,file='jet.dat',status='unknown')
      open(unit=13,file='fly.dat',status='old')
      open(unit=14,file='gndtm.inp',status='old')

c******* read in ground times **************
      do 50 i=1,7
         read(14,*) (gtm(i,j),j=1,9)
```

```
   50 continue
      close(14)

c******* read in base key ******************
      do 98 i=1,maxbases
         read(8,97,end=96) icao(i)
   98 continue
   97 format(a4)
   96 close(8)

c******* read in flytimes ****************
      do 60 i=1,maxbases**2
         read(13,*,end=61) a,b,flytm(a,b)
   60 continue
   61 close(13)

c******* read in routes ******************
      do 99 i=1,maxrts
         read(10,101,end=100) (micao(i,j),reason(i,j),j=1,maxstops)
         do 500 k=1,maxstops
            if (reason(i,k).eq.0) then
               numstop(i)=k-1
               goto 99
            endif
  500    continue
   99 continue
  100 close(10)
  101 format(50(1x,a4,i1))

      numroutes=i-1
      do 80 i=1,numroutes
         do 81 j=1,numstop(i)
            do 82 k=1,maxbases
               if (micao(i,j).eq.icao(k)) then
                  nicao(i,j)=k
               endif
   82       continue
   81    continue
   80 continue

c********** read in route frequencies & AC ********
c***** 1 = C005
c***** 2 = C141
c***** 3 = C130
c***** 4 = DC8
c***** 5 = DC10
c***** 6 = B747
c***** 7 = C17

      do 90 i=1,numroutes
         read(9,*) freq(i),actype(i)
```

```
   9C continue
      close(9)

c******* build route.dat file ******************
      do 190 i=1, numroutes
         flytot=0
         gtmtot=0
         cycle=0
         do 110 j=1, (numstop(i)-1)
            flytot = flytot + flytm(nicao(i,j),nicao(i,j+1))
            if (flytm(nicao(i,j),nicao(i,j+1)).eq.0) then
               print*,nicao(i,j),nicao(i,j+1),' no flytime'
            endif
            gtmtot = gtmtot + gtm(actype(i),reason(i,j))
  110    continue
         cycle = flytot + gtmtot
         ab = cycle * freq(i) / 720.0
         multac = int(ab+1)
         turn = multac * 720 / freq(i)
         endtime = turn - cycle
         do 120 ace=totac+1, totac+multac
            if (ace.eq.totac+1) then
               startime = origstart
            else
               startime = startime + turn/multac
            endif
            do 130 l=1,numstop(i)
               iroute(ace,l,1) = ace
               iroute(ace,l,2) = l
               iroute(ace,l,3) = nicao(i,l)
               if (l.eq.1) then
                  route(ace,l) = startime
               elseif (l.eq.numstop(i)) then
                  route(ace,l) = endtime
               else
                  route(ace,l) = gtm(actype(i),reason(i,l))
               endif
               iroute(ace,l,4) = i
               iroute(ace,l,5) = gtm(actype(i),9)
               totlines = totlines + 1
  130       continue
  120    continue
c**  In the original program the 2.5 below was 36.
         origstart = origstart + 2.5
         if (origstart.gt.336) then
            origstart = 1
         endif
         totac = totac + multac
  190 continue

c******* write the jet.dat file ******************
```

```
      do 200 i=1, totac
         write(12,1000) i,iroute(i,1,5)
  200 continue
 1000 format(2i4)

c******* write the route.dat file ***************
      do 300 i=1,totac
        do 400 j=1,numstop(iroute(i,1,4))

write(11,2000)(iroute(i,j,k),k=1,3),route(i,j),iroute(i,j,4)
  400   continue
  300 continue
 2000 format(3i5,f8.2,i5)
c*** The print statement below was added by Captain Moul at AFIT.

      print *, ' ROUTE.DAT file built, next step:  run RAWSCH'
      end
```

Appendix B. *The RAWSCH.FOR Program*

The FORTRAN program which follows takes the mission schedule shell created by the program found in Appendix A and converts it into the monthly flight schedule. This program essentially runs itself, after the other program has been run. The FORTRAN code follows:

The input files used by HQ MAC/XPYR's FORTRAN program supply nearly all the data required to create the detailed *flight* schedule. The files shown in Appendices J and K contain all information necessary to determine the time needed during each flight for flying and for ground activities. By using this data and the time at which each tail number begins its first flight (provided by HQ MAC/XPYR's scheduling program output), the *flight* scheduling program calculates the time at which *each flight* will begin during a 720-hour month. Thus, except for directional information, data in a form provided by current HQ MAC/XPYR files provides everything necessary for defining each specific flight.

The output file from the program has a number fields for each flight into which data is stored to describe the entire flight path. A flight schedule itemized in this manner serves three purposes. First, a simulation program can treat each flight as a separate entity flowing through the route system. Second, all information associated with a particular flight's stop at a particular location can be stored together with a set of fields, allowing direct computer access during simulated flight. And third, this provides a file which might be copied and manipulated by a computer at some later time to reflect a modified schedule. Schedule manipulation could then be accomplished by individual *flight*. Without this detail, as currently output from MAC's scheduling program, no manipulation of individual flights could be directly accomplished

Though this research is not intended to delve into the schedule changes which might result in better schedules, schedule adjustment might need to be made at the

detailed level. For instance, if a determination is made to start a flight earlier in the month and then hold it on the ground longer at its first stop, the times associated with these activities could be directly changed in that flight's first two repeating fields. Chapter V provides further discussion of the type of schedule adjustment for which this level of detail in a flight schedule might be mandatory.

The output file from this *flight* scheduling program only provides a one-month schedule based on the output from MAC's mission scheduling program. As just mentioned, later manipulation of this output file can be made. Furthermore, flight simulation results might be more realistic when the flight schedule covers a longer simulated period of flight operations. Therefore, to allow conversion of a monthly flight schedule of the form output from this program into a multiple-month schedule, another FORTRAN program was developed.

```
c*****  This is the RAWSCH program, which takes the ROUTE.DAT schedule
c*****  provided by the CARGRT program, and converts it to SRAW1.DAT
c*****  The SRAW1.DAT file must then be run through the MULTSCH
c*****  program for multiple months in schedule.
c*****  This program was written by Captain Moul at AFIT.
      COMMON ISTOP, MAXBAS, MAXFLT, ENDMTH
      DIMENSION FLYTIM(15,15), SCHED(999,47), TEMP(13)
      DIMENSION FLIGHTS(999), ACTYPE(999), CAP(10), FDIRCT(100,14)
      ISTOP = 47
      MAXFLT = 999
      MAXBAS = 15
      ENDMTH = 719.9999

      OPEN(UNIT=11,FILE='ROUTE.DAT',STATUS='OLD')
      OPEN(UNIT=12,FILE='FLY.DAT',STATUS='OLD')
      OPEN(UNIT=13,FILE='SRAW1.DAT',STATUS='NEW')
      OPEN(UNIT=14,FILE='FREQ.INP',STATUS='OLD')
      OPEN(UNIT=15,FILE='FDIRCT.DAT',STATUS='OLD')
      OPEN(UNIT=16,FILE='GNDTM.INP',STATUS='OLD')
      open(unit=99,file='out.out',status='unknown')

      I = 1
  200 READ(UNIT=14,FMT=*,END=210) FLIGHTS(I), ACTYPE(I)
      I = I + 1
      GOTO 200
  210 CLOSE(UNIT=14,STATUS='KEEP')

      I = 0
  220 READ(UNIT=15,FMT='(13F3.0)',END=240) (TEMP(J),J=1,13)
```

```
      I = I + 1
      DO 230, J=1, 13
         FDIRCT(I,J) = TEMP(J)
  230 CONTINUE
      GOTO 220
  240 CLOSE(UNIT=15,STATUS='KEEP')

      I = 0
  250 READ(UNIT=16,FMT=*,END=260) (TEMP(J),J=1,9)
      I = I + 1
      CAP(I) = TEMP(9)
      GOTO 250
  260 CLOSE(UNIT=16,STATUS='KEEP')

   10 READ(UNIT=12,FMT=19,END=20) ILEAVE, JARRIV, FTIME
   19 FORMAT(I4,I4,F7.2)
      FLYTIM(ILEAVE,JARRIV) = FTIME
      GOTO 10
   20 CLOSE(UNIT=12,STATUS='KEEP')

   30 READ(UNIT=11,FMT=39,END=60) IPLANE, NUMSTP, JBASE, TIME, MISSON
   39 FORMAT(I5,I5,I5,F8.2,I5)
      IF( REAL(IPLANE) .EQ. SCHED(IROW,3) ) THEN
c**  here have a matching plane number
         IF( REAL(JBASE) .EQ. SCHED(IROW,6) ) THEN
            DO 40, INDEX=11, ISTOP, 3
c**  looking for the end of a route.
               IF( SCHED(IROW,INDEX) .EQ. 0.) THEN
                  SCHED(IROW,INDEX) = REAL(JBASE)
                  CALL FLY(SCHED,IROW,FLYTIM)
                  CALL REPEAT(SCHED,IROW,TIME)
                  GOTO 30
               ENDIF
   40       CONTINUE
         ELSE
            DO 50, INDEX=11, ISTOP-3, 3
               IF( SCHED(IROW,INDEX) .EQ. 0.) THEN
                  SCHED(IROW,INDEX) = REAL(JBASE)
                  SCHED(IROW,INDEX+1) = TIME
                  GOTO 30
               ENDIF
   50       CONTINUE
         ENDIF
      ELSE
c**  here do not have a matching plane number'
         IROW = IROW + 1
         SCHED(IROW,1) = REAL(MISSON)
         SCHED(IROW,3) = REAL(IPLANE)
         SCHED(IROW,4) = CAP(INT(ACTYPE(MISSON)))
         SCHED(IROW,5) = FDIRCT(MISSON,1)
         SCHED(IROW,6) = REAL(JBASE)
```

```
         SCHED(IROW,7) = 0.
         SCHED(IROW,8) = REAL(JBASE)
         SCHED(IROW,9) = TIME
      ENDIF
      GOTO 30
   60 CLOSE(UNIT=11,STATUS='KEEP')

      IWROTE = 0
      TEMPHI = -1.
  120 TEMPLO = 100000.
      DO 130, I=1, MAXFLT
         IF( SCHED(I,1) .EQ. 0.) GOTO 130
         IF( SCHED(I,9) .LT. TEMPLO ) TEMPLO = SCHED(I,9)
         IF( SCHED(I,9) .GT. TEMPHI ) TEMPHI = SCHED(I,9)
  130 CONTINUE
      DO 140, I=1, IROW
c**  Next if assigns flight numbers, eliminates flight from further
c**  consideration, sends back to find next to take off
         IF( SCHED(I,9) .EQ. TEMPLO ) THEN
            FLTNUM = FLTNUM + 1.
            SCHED(I,2) = FLTNUM
            write(99,138) (sched(i,kk),kk=1,11)
            WRITE(UNIT=13,FMT=138) (SCHED(I,KK),KK=1,11)
  138       FORMAT(' ',F3.0,F4.0,6(F3.0),F7.2,F5.2,F3.0)
            WRITE(UNIT=13,FMT=139) (SCHED(I,KK),KK=12,29)
            WRITE(UNIT=13,FMT=139) (SCHED(I,KK),KK=30,ISTOP)
  139       FORMAT(' ',6(F5.2,F5.2,F3.0))
            IWROTE = IWROTE + 1
            IF( IWROTE .EQ. IROW ) GOTO 150
            SCHED(I,1) = 0.
            SCHED(I,9) = TEMPHI + 1.
            GOTO 120
         ENDIF
  140 CONTINUE
  150 CLOSE(UNIT=13,STATUS='KEEP')
      PRINT *, ' The SRAW1.DAT raw file is complete.'
      PRINT *, ' You can copy and edit this file for other flight',
     1' schedules.'
      PRINT *
      PRINT *, ' You must run MULTSCH to make the SCHED.DAT file',
     1' needed for SLAM.'
      close(unit=99,status='keep')
      END

c**  This subroutine fills in the flight times from one place to
c**  another for a row in the schedule
      SUBROUTINE FLY(SCHED,IROW,FLYTIM)
      COMMON ISTOP, MAXBAS, MAXFLT
      DIMENSION SCHED(MAXFLT,ISTOP), FLYTIM(MAXBAS,MAXBAS)

      DO 10, K=8, ISTOP-3, 3
```

```
         I = INT( SCHED(IROW,K) )
         J = INT( SCHED(IROW,K+3) )
         SCHED(IROW,K+2) = FLYTIM(I,J)
   10 CONTINUE
   20 RETURN
      END

c**  This subroutine creates new flights for the schedule, based on
c**  the TIME read in from the last leg of journey.
      SUBROUTINE REPEAT(SCHED,IROW,TIME)
      COMMON ISTOP, MAXBAS, MAXFLT, ENDMTH
      DIMENSION SCHED(MAXFLT,ISTOP)

      TEMP = 0.
      DO 10, K=12, ISTOP-2, 3
            TEMP = TEMP + SCHED(IROW,K) + SCHED(IROW,K+1)
   10 CONTINUE
      ADD = SCHED(IROW,10) + TEMP + TIME
   20 IF( ( SCHED(IROW,9) + ADD) .GT. ENDMTH ) GOTO 40
      IROW = IROW + 1
      DO 30, K=1, ISTOP
         SCHED(IROW,K) = SCHED(IROW-1,K)
   30 CONTINUE
      SCHED(IROW,9) = SCHFD(IROW,9) + ADD
      GOTO 20
   40 RETURN
      END
```

Appendix C. *The MULTSCH.FOR FORTRAN Program*

The MAC computer channel route model's simulation program results express cargo delay for each cargo C-D pair that might be realized if a month's schedule were in operation for a period of time longer than one month. Even though the average information might obscure individual delay-causing factors, the use of a longer period of time may help obviate bias caused by model operation without cargo flowing through the system initially. For a similar reason, this research developed a simulation relying on a multiple-month schedule.

This separate FORTRAN program produces the multiple-month schedule actually used in the simulation program. The program takes a monthly *flight* schedule and adds flights to begin at the same relative time, with the same characteristics, in subsequent months. Each subsequent month's flight schedule is a duplicate of the first -- just delayed in time by the appropriate number of hours (e.g., 720, 1440, etc.).

The program user provides the number of months the schedule will cover and the number of the schedule iteration. This research used a three-month schedule. The first iteration always uses the initial, unmodified schedule (like that created from MAC's scheduling program). Schedules which are modifications of that schedule can still use the multiple-month schedule-creation program to generate a *flight* schedule for use in simulation.

The FORTRAN program that follows can multiply any *flight* schedule of the form created by the program found in Appendix B. The user must interactively tell the computer how many months the schedule will cover. The user must also tell the computer which input file will be used in the next iteration of the SLAM program found in Appendices D and E.

The first schedule iteration uses the schedule provided by the HQ MAC/XPYR scheduling program, after it has been acted upon by the RAWSCH program (i.e., the output **SRAW1.DAT** file). A user of this proposed method might, after reviewing the results obtained from the simulation program, decide to adjust the times when flight legs occur. By copying the **SRAW1.DAT** file and adjusting the times in the copied file, a new one-month schedule could be made. For example, if simulation results were desired with the first flight commencing one hour into the month, instead of at the very beginning, a **SRAW2.DAT** could be made from **SRAW1.DAT** with this change. Then to create the multiple-month schedule for the simulation, the **MULTSCH** program will use the **SRAW2.DAT** file when the user tells the computer program that this is iteration *2*.

The output from the MULTSCH program is of the exact same form as the SRAW1.DAT file – just larger because it covers more months. This schedule is stored in the **SCHED.DAT** file, which is one of the major input files required to run the simulation program proposed by this research. The FORTRAN code follows:

```
c*****  This is the MULTSCH program.  This program takes a SCH##RAW
c*****  file and adds additional flights to provide a multiple-month
c*****  schedule to run the SLAM cargo simulation.  The output file
c*****  of this program is SCHED.DAT, which is used by the SLAM code.
c*****  This program was written by Captain Moul at AFIT.
c*****  NOTE:  On first iteration, using MAC's CARGRT program, through
c*****  the RAWSCH program the input file would be SCH1RAW.DAT  After
c*****  that, the modeler can copy and edit the SCH1RAW.DAT file into
c*****  a SCH2RAW.DAT, etc.  Then SLAM runs can be accomplished using
c*****  a modified flight schedule.

      DIMENSION SCHED(999,47)
      CHARACTER*20 ITER, TEMP, FILNAM
      NATFLT = 47
      PRINT *, ' What SCHED iteration is this ? '
      READ(*,'(A20)') ITER
      J = LEN(ITER)
      FILNAM = 'SRAW' // ITER(1:J)
      PRINT *
      PRINT *, 'How many months to be in schedule ?'
      READ(*,'(I2)') MONTHS

      OPEN(UNIT=11,FILE=FILNAM,STATUS='OLD')
```

```
      OPEN(UNIT=12,FILE='SCHED.DAT',STATUS='UNKNOWN')
      open(unit=99,file='out.out',status='unknown')

      DO 10, I=1, 999
         IROW = IROW + 1
         READ(UNIT=11,FMT=8,END=20) (SCHED(I,KK),KK=1,11)
    8    FORMAT(1X,F3.0,F4.0,6(F3.0),F7.2,F5.2,F3.0)
         READ(UNIT=11,FMT=9,END=20) (SCHED(I,KK),KK=12,29)
         READ(UNIT=11,FMT=9,END=20) (SCHED(I,KK),KK=30,NATFLT)
    9    FORMAT(1X,6(F5.2,F5.2,F3.0))
   10 CONTINUE
   20 CLOSE(UNIT=11,STATUS='KEEP')
      IROW = IROW - 1

C**  Next set of lines multiply schedule by number of months input.
C**  Note: same flights in other months will occur at same relative
C**  time in month - leaving discontinuity between months.
      DO 30, I=1, IROW
         FLTNUM = FLTNUM + 1.
         SCHED(I,2) = FLTNUM
         WRITE(UNIT=12,FMT=28) (SCHED(I,KK),KK=1,11)
   28    FORMAT(' ',F3.0,F4.0,6(F3.0),F7.2,F5.2,F3.0)
         WRITE(UNIT=12,FMT=29) (SCHED(I,KK),KK=12,29)
         WRITE(UNIT=12,FMT=29) (SCHED(I,KK),KK=30,NATFLT)
   29    FORMAT(' ',6(F5.2,F5.2,F3.0))
   30 CONTINUE
      DO 50, M=2, MONTHS
         DO 40, I=1, IROW
            FLTNUM = FLTNUM + 1.
            SCHED(I,2) = FLTNUM
            SCHED(I,9) = SCHED(I,9) + 720.
            WRITE(UNIT=12,FMT=28) (SCHED(I,KK),KK=1,11)
            WRITE(UNIT=12,FMT=29) (SCHED(I,KK),KK=12,29)
            WRITE(UNIT=12,FMT=29) (SCHED(I,KK),KK=30,NATFLT)
   40    CONTINUE
   50 CONTINUE
      PRINT *
      PRINT *, 'The SCHED.DAT file is now ready for running SLAM.'
      PRINT *
      PRINT *, 'The SLAM and fortran codes are both CARGO.'
      CLOSE(UNIT=12,STATUS='KEEP')
      close(unit=99,status='keep')
      END
```

Appendix D. *The CARGO.DAT SLAM Code File*

The SLAM code which follows is all that is necessary to run the cargo simulation, along with the FORTRAN program inserts found in Appendix E. To modify this program the following changes might need to be made:

- *LIMITS* - This research's simulation is unusual because of the requirement to maintain attribute values for *all* of these entities. For this reason, this research follows Pritsker's advice of "the judicious use of a safety factor" when establishing "the total number of entities that can exist in the model at one time" (18:268).
- *INIT* - start time and end time in hours (time schedule covers).
- *CREATEs* - need to set up a CREATE portion for each O-D cargo pair, with mission flag, origin, destination, direction, and current location, .
- QUEUEs - one numbered QUEUE for each airport, renumber HOLD, TEMP, and DONE above the highest airport number.

```
GEN,CAPTMOUL,CARGO DELIVERY,4/30/92,1,N,N,Y/Y,N,Y/1,72;
LIMITS,21,98,10000;   files(queues),attributes,entities in file
INIT,0,2160;             go three months, by hour
NETWORK;
;
 C1t4 CREATE,,,,,;              create one 1to4 cargo
      ASSIGN, ATRIB(1)=-1., ATRIB(2)=1., ATRIB(3)=4.,
              ATRIB(4)=5., ATRIB(7)=1.,;
      ACT,USERF(5),,;           split for next cargo
      GOON/2;
      ACT,,,C1t4;
      ACT/5,,,Q001;             enter 1to4 cargo
 C1t6 CREATE,,,,,;              create one 1to6 cargo
      ASSIGN, ATRIB(1)=-1., ATRIB(2)=1., ATRIB(3)=6.,
              ATRIB(4)=5., ATRIB(7)=1.;
      ACT,USERF(5),,;           split for next cargo
      GOON/2;
      ACT,,,C1t6;
      ACT/6,,,Q001;             enter 1to6 cargo
 C1t9 CREATE,,,,,;              create one 1to9 cargo
```

```
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=1., ATRIB(3)=9.,
             ATRIB(4)=5., ATRIB(7)=1.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C1t9;
     ACT/7,,,Q001;             enter 1to9 cargo
C1tW CREATE,,,,,;              create one 1to12 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=1., ATRIB(3)=12.,
             ATRIB(4)=5., ATRIB(7)=1.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C1tW;
     ACT/8,,,Q001;             enter 1to12 cargo
C2t3 CREATE,,,,,;              create one 2to3 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=3.,
             ATRIB(4)=5., ATRIB(7)=2.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C2t3;
     ACT/9,,,Q002;             enter 2to3 cargo
C2t5 CREATE,,,,,;              create one 2to5 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=5.,
             ATRIB(4)=5., ATRIB(7)=2.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C2t5;
     ACT/10,,,Q002;            enter 2to5 cargo
C2t7 CREATE,,,,,;              create one 2to7 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=7.,
             ATRIB(4)=5., ATRIB(7)=2.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C2t7;
     ACT/11,,,Q002;            enter 2to7 cargo
C2t8 CREATE,,,,,;              create one 2to8 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=8.,
             ATRIB(4)=5., ATRIB(7)=2.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C2t8;
     ACT/12,,,Q002;            enter 2to8 cargo
C2t0 CREATE,,,,,;              create one 2to10 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=10.,
             ATRIB(4)=5., ATRIB(7)=2.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C2t0;
     ACT/13,,,Q002;            enter 2to10 cargo
C2tW CREATE,,,,,;              create one 2to12 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=2., ATRIB(3)=12.,
             ATRIB(4)=5., ATRIB(7)=2.;
```

```
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C2tW;
     ACT/14,,,Q002;           enter 2to12 cargo
C3t2 CREATE,,,,,;             create one 3to2 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=3., ATRIB(3)=2.,
             ATRIB(4)=15., ATRIB(7)=3.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C3t2;
     ACT/15,,,Q003;           enter 3to2 cargo
C3t4 CREATE,,,,,;             create one 3to4 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=3., ATRIB(3)=4.,
             ATRIB(4)=5., ATRIB(7)=3.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C3t4;
     ACT/16,,,Q003;           enter 3to4 cargo
C3t6 CREATE,,,,,;             create one 3to6 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=3., ATRIB(3)=6.,
             ATRIB(4)=5., ATRIB(7)=3.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C3t6;
     ACT/17,,,Q003;           enter 3to6 cargo
C3t0 CREATE,,,,,;             create one 3to10 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=3., ATRIB(3)=10.,
             ATRIB(4)=5., ATRIB(7)=3.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C3t0;
     ACT/18,,,Q003;           enter 3to10 cargo
C4t1 CREATE,,,,,;             create one 4to1 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=4., ATRIB(3)=1.,
             ATRIB(4)=15., ATRIB(7)=4.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C4t1;
     ACT/19,,,Q004;           enter 4to1 cargo
C4t6 CREATE,,,,,;             create one 4to6 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=4., ATRIB(3)=6.,
            ATRIB(4)=5., ATRIB(7)=4.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C4t6;
     ACT/20,,,Q004;           enter 4to6 cargo
C4t9 CREATE,,,,,;             create one 4to9 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=4., ATRIB(3)=9.,
             ATRIB(4)=5., ATRIB(7)=4.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
```

```
     ACT,,,C4t9;
     ACT/21,,,Q004;            enter 4to9 cargo
C4t0 CREATE,,,,,;              create one 4to10 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=4., ATRIB(3)=10.,
             ATRIB(4)=5., ATRIB(7)=4.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C4t0;
     ACT/22,,,Q004;            enter 4to10 cargo
C5t2 CREATE,,,,,;              create one 5to2 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=5., ATRIB(3)=2.,
             ATRIB(4)=15., ATRIB(7)=5.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C5t2;
     ACT/23,,,Q005;            enter 5to2 cargo
C5t4 CREATE,,,,,;              create one 5to4 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=5., ATRIB(3)=4.,
             ATRIB(4)=15., ATRIB(7)=5.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C5t4;
     ACT/24,,,Q005;            enter 5to4 cargo
C5t7 CREATE,,,,,;              create one 5to7 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=5., ATRIB(3)=7.,
             ATRIB(4)=5., ATRIB(7)=5.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C5t7;
     ACT/25,,,Q005;            enter 5to7 cargo
C5t8 CREATE,,,,,;              create one 5to8 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=5., ATRIB(3)=8.,
             ATRIB(4)=5., ATRIB(7)=5.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C5t8;
     ACT/26,,,Q005;            enter 5to8 cargo
C5tW CREATE,,,,,;              create one 5t12 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=5., ATRIB(3)=12.,
             ATRIB(4)=5., ATRIB(7)=5.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C5tW;
     ACT/27,,,Q005;            enter 5to12 cargo
C6t1 CREATE,,,,,;              create one 6to1 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=6., ATRIB(3)=1.,
             ATRIB(4)=15., ATRIB(7)=6.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C6t1;
     ACT/28,,,Q006;            enter 6to1 cargo
```

```
C6t3 CREATE,,,,,;              create one 6to3 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=6., ATRIB(3)=3.,
             ATRIB(4)=15., ATRIB(7)=6.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C6t3;
     ACT/29,,,Q006;            enter 6to3 cargo
C6t4 CREATE,,,,,;              create one 6to4 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=6., ATRIB(3)=4.,
             ATRIB(4)=15., ATRIB(7)=6.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C6t4;
     ACT/30,,,Q006;            enter 6to4 cargo
C6t9 CREATE,,,,,;              create one 6to9 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=6., ATRIB(3)=9.,
             ATRIB(4)=10., ATRIB(7)=6.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C6t9;
     ACT/31,,,Q006;            enter 6to9 cargo
C6t0 CREATE,,,,,;              create one 6to10 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=6., ATRIB(3)=10.,
             ATRIB(4)=10., ATRIB(7)=6.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C6t0;
     ACT/32,,,Q006;            enter 6to10 cargo
C7t2 CREATE,,,,,;              create one 7to2 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=7., ATRIB(3)=2.,
             ATRIB(4)=10., ATRIB(7)=7.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C7t2;
     ACT/33,,,Q007;            enter 7to2 cargo
C7t5 CREATE,,,,,;              create one 7to5 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=7., ATRIB(3)=5.,
             ATRIB(4)=10., ATRIB(7)=7.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C7t5;
     ACT/34,,,Q007;            enter 7to5 cargo
C7t8 CREATE,,,,,;              create one 7to8 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=7., ATRIB(3)=8.,
             ATRIB(4)=5., ATRIB(7)=7.;
     ACT,USERF(5),,;           split for next cargo
     GOON/2;
     ACT,,,C7t8;
     ACT/35,,,Q007;            enter 7to8 cargo
C8t2 CREATE,,,,,;              create one 8to2 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=8., ATRIB(3)=2.,
```

```
            ATRIB(4)=5., ATRIB(7)=8.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C8t2;
     ACT/36,,,Q008;           enter 8to2 cargo
C8t4 CREATE,,,,,;             create one 8to4 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=8., ATRIB(3)=4.,
            ATRIB(4)=5., ATRIB(7)=8.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C8t4;
     ACT/37,,,Q008;           enter 8to4 cargo
C8t5 CREATE,,,,,;             create one 8to5 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=8., ATRIB(3)=5.,
            ATRIB(4)=5., ATRIB(7)=8.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C8t5;
     ACT/38,,,Q008;           enter 8to5 cargo
C8t7 CREATE,,,,,;             create one 8to7 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=8., ATRIB(3)=7.,
            ATRIB(4)=5., ATRIB(7)=8.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C8t7;
     ACT/39,,,Q008;           enter 8to7 cargo
C9t1 CREATE,,,,,;             create one 9to1 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=9., ATRIB(3)=1.,
            ATRIB(4)=15., ATRIB(7)=9.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C9t1;
     ACT/40,,,Q009;           enter 9to1 cargo
C9t4 CREATE,,,,,;             create one 9to4 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=9., ATRIB(3)=4.,
            ATRIB(4)=10., ATRIB(7)=9.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C9t4;
     ACT/41,,,Q009;           enter 9to4 cargo
C9t6 CREATE,,,,,;             create one 9to6 cargo
     ASSIGN, ATRIB(1)=-1., ATRIB(2)=9., ATRIB(3)=6.,
            ATRIB(4)=10., ATRIB(7)=9.;
     ACT,USERF(5),,;          split for next cargo
     GOON/2;
     ACT,,,C9t6;
     ACT/42,,,Q009;           enter 9to6 cargo
C0t1 CREATE,,,,,;             create one 10to1 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=10., ATRIB(3)=1.,
            ATRIB(4)=10., ATRIB(7)=10.;
     ACT,USERF(5),,;          split for next cargo
```

```
     GOON/2;
     ACT,,,C0t1;
     ACT/43,,,Q010;              enter 10to1 cargo
C0t3 CREATE,,,,,;                create one 10to3 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=10., ATRIB(3)=3.,
             ATRIB(4)=10., ATRIB(7)=10.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,C0t3;
     ACT/44,,,Q010;              enter 10to3 cargo
C0t4 CREATE,,,,,;                create one 10to4 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=10., ATRIB(3)=4.,
             ATRIB(4)=10., ATRIB(7)=10.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,C0t4;
     ACT/45,,,Q010;              enter 10to4 cargo
C0t6 CREATE,,,,,;                create one 10to6 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=10., ATRIB(3)=6.,
             ATRIB(4)=10., ATRIB(7)=10.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,C0t6;
     ACT/46,,,Q010;              enter 10to6 cargo
CWt1 CREATE,,,,,;                create one 12to1 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=12., ATRIB(3)=1.,
             ATRIB(4)=15., ATRIB(7)=12.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,CWt1;
     ACT/48,,,Q012;              enter 12to1 cargo
CWt2 CREATE,,,,,;                create one 12to2 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=12., ATRIB(3)=2.,
             ATRIB(4)=15., ATRIB(7)=12.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,CWt2;
     ACT/49,,,Q012;              enter 12to2 cargo
CWt5 CREATE,,,,,;                create one 12to5 cargo
     ASSIGN, ATRIB(1)=0., ATRIB(2)=12., ATRIB(3)=5.,
             ATRIB(4)=15., ATRIB(7)=12.;
     ACT,USERF(5),,;             split for next cargo
     GOON/2;
     ACT,,,CWt5;
     ACT/50,,,Q012;              enter 12to5 cargo
JETS CREATE,,,,,;                create one plane/flight
INFO ACT,USERF(1),,;             split for next flight
     GOON/2;
     ACT,0.,ATRIB(1).GT.0.,JETS;    create next plane
DPRT ACT/1,0.,ATRIB(1).GT.0.,;        plane departs
FILL EVENT,1,1;              looks at onboard cargo
```

```
 PNEW EVENT,2,1;              gets new cargo
  FLY ACT/2,USERF(2),;        fly to next stop
      ASSIGN, ATRIB(7)=ATRIB(7)+1.;  increase leg counter
      ACT,0.,1.,DROP;
 DROP EVENT,3,1;             update place (hold)
 FDIR EVENT,4,1;             plane direction change ?
 CDIR EVENT,5,1;             cargo direction change ?
 DOFF EVENT,6,1;             cargo getting off ?
 FFIN EVENT,7,1;             last stop of plane
      GOON/1;
  SIT ACT/3,USERF(4),USERF(3).NE.ATRIB(6),FILL;  sit on ground
;    userf(4) returns groundtime, userf(3) returns current stop
 LAST ACT/4,0,USERF(3).EQ.ATRIB(6),;              at last stop
      TERM;
 Q001 QUEUE(1),0,,,;
 Q002 QUEUE(2),0,,,;
 Q003 QUEUE(3),0,,,;
 Q004 QUEUE(4),0,,,;
 Q005 QUEUE(5),0,,,;
 Q006 QUEUE(6),0,,,;
 Q007 QUEUE(7),0,,,;
 Q008 QUEUE(8),0,,,;
 Q009 QUEUE(9),0,,,;
 Q010 QUEUE(10),0,,,;
 Q011 QUEUE(11),0,,,;
 Q012 QUEUE(12),0,,,;
 TEMP QUEUE(19),0,,,;
 HOLD QUEUE(20),0,,,;
 DONE QUEUE(21),0,,,;
      ACT(1),0.,1.,;
 CARG EVENT,8,1;
      TERM;
      ENDNETWORK;
FIN;
```

Appendix E. *The CARGO.FOR File for SLAM Inserts*

```
      PROGRAM MAIN
      DIMENSION NSET(3000000)
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)
      COMMON QSET(3000000)
      EQUIVALENCE (NSET(1),QSET(1))
      NNSET=3000000
      NCRDR=5
      NPRNT=6
      NTAPE=7
      NPLOT=2
C** Next lines to pass common information to subroutines.  These need
C** to be adjusted for changes to network (e.g., number of airports,
C** number of cargo or aircraft attributes).  Also, for all routines
C** the COMMON block matrix dimensions must be adjusted when needed.
C** Current limitations: 12 legs/flight, 18 stops/ton of cargo.
      NUMQUE = 12
      NATCAR = 98
      NATFLT = 47
      ENDAT = 720.
      BSTATS = 720.
      ESTATS = 1419.9999
      CALL SLAM
      STOP
      END

C**  This subroutine takes care of initial activities required, such
C**  as initial opening of files for reading and writing.
      SUBROUTINE INTLC
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
```

```
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)
      DIMENSION TEMP(14)

      OPEN(UNIT=1,FILE='SCHED.DAT',STATUS='OLD')
      OPEN(UNIT=3,FILE='DEMAND.DAT',STATUS='OLD')
      OPEN(UNIT=4,FILE='CDIRCT.DAT',STATUS='OLD')
      OPEN(UNIT=8,FILE='LEGS.OUT',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE='QUEUES.OUT',STATUS='UNKNOWN')
      OPEN(UNIT=10,FILE='CMSSION.DAT',STATUS='OLD')
      OPEN(UNIT=11,FILE='FDIRCT.DAT',STATUS='OLD')
      OPEN(UNIT=30,FILE='FLEGS.TMP',STATUS='UNKNOWN')
      OPEN(UNIT=31,FILE='EVERY.TMP',STATUS='UNKNOWN')
      OPEN(UNIT=99,FILE='OUT.OUT',STATUS='UNKNOWN')

c  The next lines read the cargo demand, direction changes, and
c  mission number changes between the origin-destination pairs and
c  converts it into internal matrix form more easily used later
c  (in USERF(5) for cargo demand generation and in CDIR for
c  DIRECTion and MiSSION changes).

  100 READ(UNIT=3,FMT='(2(I4),F10.2)',END=110) IREAD1, IREAD2, READ3
      IF( READ3 .EQ. 0.) READ3 = .000000001
      BTWEEN = ENDAT / READ3
      DEMAND(IREAD1,IREAD2) = BTWEEN
      GOTO 100
  110 CLOSE(UNIT=3,STATUS='KEEP')

  120 READ(UNIT=4,FMT='(2(I4),F6.0)',END=130) IREAD1, IREAD2, READ3
      CDIRCT(IREAD1,IREAD2) = READ3
      GOTO 120
  130 CLOSE(UNIT=4,STATUS='KEEP')

  140 READ(UNIT=10,FMT='(5(F4.0))',END=160) (TEMP(KK),KK=1,5)
      I = I + 1
      DO 150, J=1, 5
         CMSION(I,J) = TEMP(J)
  150 CONTINUE
      GOTO 140
  160 CLOSE(UNIT=10,STATUS='KEEP')

c  Now we read in the stops where aircraft are changing directions
c  into the FDIRCT matrix for later use in DDIR.
  170 READ(UNIT=11,FMT='(13(F3.0))',END=190) (TEMP(KK),KK=1,13)
      IT = IT + 1
      DO 180, J=1,13
         FDIRCT(IT,J) = TEMP(J)
  180 CONTINUE
      GOTO 170
  190 CLOSE(UNIT=11,STATUS='KEEP')

      RETURN
```

```
      END

C**  Here is the beginning of all the events.  Note that the events
C**  really only directly affect aircraft in the SLAM code.  The cargo
C**  is adjusted by the swapping of attribute values indirectly
C**  through the sequence of events.
      SUBROUTINE EVENT(I)
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)
      DIMENSION A(100), EVERY(8), FLEGS(9)

      GOTO(1000,2000,3000,4000,5000,6000,7000,8000) I

C**  This is FILL.  Fill in cargo departure time, next stop,
c**  arrival time at next stop for cargo already onboard.
 1000 CONTINUE
      JJ = INT( 3 * ATRIB(7) )

 1010 NEXT = MMFE(20)
      IF (NEXT .EQ. 0) GOTO 1030
      CALL RMOVE(-NEXT,20,A)

c  find cargo on plane, adjust cargo attributes to reflect cargo now
c  at new location, put back in HOLD.  Note that cargo dropping
c  occurs later.
      IF( A(5) .EQ. ATRIB(2) ) THEN
         DO 1020, K=9, NATCAR-4, 5
            IF ( A(K) .EQ. 0.) THEN
               A(K-1) = TNOW
               A(K) = ATRIB(2)
               A(K+1) = ATRIB(3)
               A(K+2) = ATRIB(JJ + 11)
               A(K+3) = TNOW + ATRIB(JJ + 10)
               CALL FFILE(19,A)
               GOTO 1010
            ENDIF
 1020    CONTINUE
      ELSE
         CALL FFILE(19,A)
         GOTO 1010
      ENDIF

 1030 NEXT = MMFE(19)
```

```
      IF( NEXT .EQ. 0 ) GOTO 1040
      CALL RMOVE(-NEXT,19,A)
      CALL FFILE(20,A)
      GOTO 1030

 1040 CONTINUE
      RETURN

C** This is PNEW, which looks for cargo at the current location of
C** the aircraft to be picked up, picks it up when appropriate,
C** adjusts attributes of cargo to reflect getting on plane, and
C** puts such cargo in HOLD status.
 2000 CONTINUE
      JJ = INT( 3 * ATRIB(7) )
      NOWAT = INT( ATRIB(JJ + 8) )
      IF( ATRIB(4) - CARGON(INT(ATRIB(2))) .EQ. 0. ) GOTO 2070

c  look at current location for cargo to pick up.
 2010 NEXT = MMFE(NOWAT)
      IF(NEXT .EQ. 0) GOTO 2060
      CALL RMOVE(-NEXT,NOWAT,A)
c  Note:  Next IF compares directions and allows a range for allowable
c  assignment.  Modification might expand on this decision logic.
      IF( (A(4) .GE. ATRIB(5)-5.) .AND.
     1    (A(4) .LE. ATRIB(5)+5.)     ) THEN
         IF( A(1) .EQ. 0.) THEN
            DO 2020, K=9, NATCAR-4, 5
               IF ( A(K) .EQ. 0.) THEN
                  A(5) = ATRIB(2)
                  A(6) = ATRIB(3)
                  A(K-1) = TNOW
                  A(K) = ATRIB(2)
                  A(K+1) = ATRIB(3)
                  A(K+2) = ATRIB(JJ + 11)
                  A(K+3) = TNOW + ATRIB(JJ + 10)
                  CALL FFILE(20,A)
                  CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) + 1.
                  IF( ATRIB(4)-CARGON(INT(ATRIB(2))) .EQ. 0.) GOTO 2060
                  GOTO 2010
               ENDIF
 2020       CONTINUE
         ELSE
            IROW = 0
            DO 2030, I=1, NUMQUE**2
               IROW = IROW + 1
               IF( (A(7) .EQ. CMSION(I,1)) .AND.
     1             (A(3) .EQ. CMSION(I,2))     ) GOTO 2040
 2030       CONTINUE
 2040       IF( (ATRIB(1) .EQ. CMSION(IROW,3)) .OR.
     1          (ATRIB(1) .EQ. CMSION(IROW,4)) .OR.
     2          (ATRIB(1) .EQ. CMSION(IROW,5)) ) THEN
```

```
                DO 2050, K=9, NATCAR-4, 5
                   IF ( A(K) .EQ. 0.) THEN
                      A(5) = ATRIB(2)
                      A(6) = ATRIB(3)
                      A(K-1) = TNOW
                      A(K) = ATRIB(2)
                      A(K+1) = ATRIB(3)
                      A(K+2) = ATRIB(JJ + 11)
                      A(K+3) = TNOW + ATRIB(JJ + 10)
                      CALL FFILE(20,A)
                      CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) + 1.
                      IF(ATRIB(4)-CARGON(INT(ATRIB(2))).EQ.0.) GOTO 2060
                      GOTO 2010
                   ENDIF
 2050           CONTINUE
              ELSE
                 CALL FFILE(19,A)
              ENDIF
           ENDIF
        ELSE
           CALL FFILE(19,A)
        ENDIF
        GOTO 2010

 2060 NEXT = MMFE(19)
        IF( NEXT .EQ. 0 ) GOTO 2070
        CALL RMOVE(-NEXT,19,A)
        CALL FFILE(NOWAT,A)
        GOTO 2060

 2070 CONTINUE
        RETURN

C** Dropping off cargo.  This subroutine will have to find out whether
C** cargo is destined for the next plane stop, whether it will stay on
C** the plane, or whether it is getting off (into the queue) here.
C** Cargo at destination might be problem - will it get to top of FIFO
C** list to process to reporting block ?  Can entity be inserted at
C** top of queue ?

C****** this is DROP
 3000 CONTINUE
        IF( INT(ATRIB(7)) .GT. MAXSTP(INT(ATRIB(1))) ) THEN
           MAXSTP(INT(ATRIB(1))) = INT(ATRIB(7))
        ENDIF
        JJ = INT ( 3 * ATRIB(7) )
        PLACE = ATRIB(JJ + 8)
        NOWAT = INT( ATRIB(JJ + 8) )

c  updating cargo location
 3010 NEXT = MMFE(20)
```

```
      IF (NEXT .EQ. 0) GOTO 3060
      CALL RMOVE(-NEXT,20,A)
      IF ( A(5) .EQ. ATRIB(2) ) THEN
         IF( (TNOW .GE. BSTATS) .AND. (TNOW .LT. ESTATS) ) THEN
            ILEGS = ILEGS + 1
            FLEGS(1) = ATRIB(1)
            FLEGS(2) = ATRIB(2)
            FLEGS(3) = ATRIB(3)
            FLEGS(4) = ATRIB(4)
            FLEGS(5) = ATRIB(7)
            FLEGS(6) = A(2)
            FLEGS(7) = A(3)
c  This next loop finds the time the cargo got on.
            DO 3020, I=9, NATCAR-4, 5
               IF( A(I) .EQ. ATRIB(2) ) THEN
                  IGOTON = I
                  GOTO 3030
               ENDIF
 3020       CONTINUE
 3030       FLEGS(8) = A(IGOTON-1)
            FLEGS(9) = TNOW
            WRITE(UNIT=30,FMT=*) (FLEGS(KK),KK=1,9)
         ENDIF
C** update place, put at destination or back onboard.
         A(7) = PLACE
         IF ( A(7) .EQ. A(3) ) THEN
            DO 3040, I=8, NATCAR-5, 5
               IF( A(I) .EQ. 0.) THEN
                  A(I) = TNOW
                  GOTO 3050
               ENDIF
 3040       CONTINUE
c  cargo at destination, put in DONE.
 3050       CALL FILEM(21,A)
            CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) - 1.
         ELSE
c  cargo not at destination, put back in HOLD.
            CALL FFILE(19,A)
         ENDIF
      ELSE
         CALL FFILE(19,A)
      ENDIF
      GOTO 3010

 3060 NEXT = MMFE(19)
      IF( NEXT .EQ. 0 ) GOTO 3070
      CALL RMOVE(-NEXT,19,A)
      CALL FFILE(20,A)
      GOTO 3060

 3070 CONTINUE
```

```
      RETURN

C**  This is FDIR. Check to see if plane direction needs to change.
 4000 CONTINUE
      ATRIB(5) = FDIRCT(INT(ATRIB(1)),INT(ATRIB(7))+1)
      RETURN

C** This is CDIR which checks to see if cargo direction and/or mssion
C** needs to change.  Note that all cargo, even if changing, is put
C** back in HOLD.  It is taken out, if necessary, in the next event.
 5000 CONTINUE
 5010 NEXT = MMFE(20)
      IF ( NEXT .EQ. 0) GOTO 5040
      CALL RMOVE(-NEXT,20,A)
      IF ( A(5) .EQ. ATRIB(2) ) THEN

c  The next lines check and change direction and mission,
c  respectively, as input in the CDIRCT.DAT and MSSION.DAT files.
c  This is based on current location and final destination.
         A(4) = CDIRCT( INT(A(7)), INT(A(3)) )
         IROW = 0
         DO 5020, I=1, NUMQUE**2
            IROW = IROW + 1
            IF( (A(7) .EQ. CMSION(IROW,1)) .AND.
     1          (A(3) .EQ. CMSION(IROW,2))     ) GOTO 5030
 5020    CONTINUE
 5030    IF( (CMSION(IROW,3) .EQ. 0.) .AND.
     1       (CMSION(IROW,4) .EQ. 0.) .AND.
     2       (CMSION(IROW,5) .EQ. 0.)     ) THEN
            A(1) = 0.
         ELSE
            A(1) = -1.
         ENDIF
         CALL FFILE(19,A)
      ELSE
         CALL FFILE(19,A)
      ENDIF
      GOTO 5010

 5040 NEXT = MMFE(19)
      IF( NEXT .EQ. 0 ) GOTO 5050
      CALL RMOVE(-NEXT,19,A)
      CALL FFILE(20,A)
      GOTO 5040

 5050 CONTINUE
      RETURN

C** This is DOFF, which searches for other cargo getting off for some
C** reason.  Note that both plane and cargo directions could have been
C** changed in previous events.
```

```
 6000 CONTINUE
      JJ = INT( 3 * ATRIB(7) )
      PLACE = ATRIB(JJ + 8)
      NOWAT = INT( ATRIB(JJ + 8) )
      IF ( PLACE .EQ. ATRIB(6) ) RETURN

c   checking hold for other cargo getting off
 6010 NEXT = MMFE(20)
      IF ( NEXT .EQ. 0 ) GOTO 6040
      CALL RMOVE(-NEXT,20,A)
      IF ( A(5) .EQ. ATRIB(2) ) THEN

c*** if going right direction, AND, mission same as plane or not 0,
c*** will stay on the plane.
c  Note:  Next IF compares directions and allows a range for allowable
c  assignment.  Modification might expand on this decision logic.
         IF( (A(4) .GE. ATRIB(5)-5.) .AND.
     1       (A(4) .LE. ATRIB(5)+5.)     ) THEN
            IF( A(1) .NE. 0.) THEN
               IROW = 0
               DO 6020, I=1, NUMQUE*NUMQUE
                  IROW = IROW + 1
                  IF( (A(7) .EQ. CMSION(IROW,1)) .AND.
     2                (A(3) .EQ. CMSION(IROW,2))     ) GOTO 6030
 6020             CONTINUE
 6030             IF( (ATRIB(1) .EQ. CMSION(IROW,3)) .OR.
     1               (ATRIB(1) .EQ. CMSION(IROW,4)) .OR.
     2               (ATRIB(1) .EQ. CMSION(IROW,5))     ) THEN
                  CALL FFILE(19,A)
               ELSE
                  CALL FFILE(NOWAT,A)
                  CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) - 1.
               ENDIF
            ELSE
               CALL FFILE(19,A)
            ENDIF

c  otherwise will get off plane at current stop.
         ELSE
            CALL FFILE(NOWAT,A)
            CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) - 1.
         ENDIF
      ELSE
         CALL FFILE(19,A)
      ENDIF
      GOTO 6010

 6040 NEXT = MMFE(19)
      IF( NEXT .EQ. 0 ) GOTO 6050
      CALL RMOVE(-NEXT,19,A)
      CALL FFILE(20,A)
```

```
      GOTO 6040

 6050 CONTINUE
      RETURN

C** This is FFIN, which takes care of the situation when a flight is at
C** the last stop (the plane arrives at home base).  All cargo in HOLD
C** on the plane is offloaded at current location.
 7000 CONTINUE
      JJ = INT( 3 * ATRIB(7) )
      PLACE = ATRIB(JJ + 8)
      NOWAT = INT( ATRIB(JJ + 8) )
      IF (PLACE .NE. ATRIB(6) ) RETURN

 7010 NEXT = MMFE(20)
      IF (NEXT .EQ. 0) GOTO 7020
      CALL RMOVE(-NEXT,20,A)
      IF ( A(5) .EQ. ATRIB(2) ) THEN
         CALL FFILE(NOWAT,A)
         CARGON(INT(ATRIB(2))) = CARGON(INT(ATRIB(2))) - 1.
      ELSE
         CALL FFILE(19,A)
      ENDIF
      GOTO 7010

 7020 NEXT = MMFE(19)
      IF( NEXT .EQ. 0 ) GOTO 7030
      CALL RMOVE(-NEXT,19,A)
      CALL FFILE(20,A)
      GOTO 7020

 7030 CONTINUE
      RETURN

c  This is CARG, for writing cargo atributes to common matrix EVERY.
c  Note:  unlike other above routines, this one is using attributes
c  of the cargo entities.
 8000 CONTINUE
      IF( (TNOW .GE. BSTATS) .AND. (TNOW .LT. ESTATS) ) THEN
         IEVERY = IEVERY + 1
         DO 8010, I=9, NATCAR-4, 5
            EVERY(1) = ATRIB(2)
            EVERY(2) = ATRIB(3)
            IF( I .EQ. 9) THEN
               EVERY(3) = ATRIB(2)
            ELSE
               EVERY(3) = ATRIB(I-3)
            ENDIF
            EVERY(4) = ATRIB(I)
            EVERY(5) = ATRIB(I+1)
            EVERY(6) = ATRIB(I+4)
```

```
            EVERY(7) = ATRIB(I-1)
            EVERY(8) = ATRIB(I+4) - ATRIB(I-1)
            WRITE(UNIT=31,FMT=*) (EVERY(KK),KK=1,8)
 8010    CONTINUE
      ENDIF
      RETURN

      END

C***  Begin user functions
      FUNCTION USERF(I)
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)

      GOTO(1,2,3,4,5) I

C**  Read attribute values, adjust depart time to time from TNOW.
    1 CONTINUE
      READ(UNIT=1,FMT=8,END=10) (ATRIB(J),J=1,11)
    8 FORMAT(1x,F3.0,F4.0,6(F3.0),F7.2,F5.2,F3.0)
      READ(UNIT=1,FMT=9,END=10) (ATRIB(J),J=12,29)
      READ(UNIT=1,FMT=9,END=10) (ATRIB(J),J=30,NATFLT)
    9 FORMAT(1x,6(F5.2,F5.2,F3.0))
      IF( (TNOW .GE. BSTATS) .AND. (TNOW .LT. ESTATS) ) THEN
         NUMFLT = NUMFLT + 1
      ENDIF
      NFLOWN = NFLOWN + 1
      FLTMSN(NFLOWN) = ATRIB(1)
      IF( INT(ATRIB(1)) .GT. IMSSNS) IMSSNS = INT(ATRIB(1))
      USERF = ATRIB(9) - TNOW
      RETURN

C**  End-of-file.  Kill aircraft creation sequence.

   10 ATRIB(1) = 0
      USERF = 0
      CLOSE(UNIT=1,STATUS='KEEP')
      RETURN

C** Figuring flying time to next stop.
    2 CONTINUE
      J = INT( 3. * ATRIB(7) + 10.)
      USERF = ATRIB(J)
```

```
      RETURN

C** Figuring what the current stop location is.
    3 CONTINUE
      J = INT( 3. * ATRIB(7) + 8.)
      USERF = ATRIB(J)
      RETURN

C** Figuring ground time till next departure.
    4 CONTINUE
      J = INT( 3. * ATRIB(7) + 9.)
      USERF = ATRIB(J)
      RETURN

C** Get values for time between creation for cargo.
    5 CONTINUE
      IROW = INT( ATRIB(2) )
      ICOL = INT( ATRIB(3) )
      USERF = DEMAND(IROW,ICOL)

      RETURN
      END

C**  Final subroutine called by SLAM at end of simulation.  Includes
C**  calculation and reporting of delay figures (through called
C**  subroutines using data passed through common EVERY information.
C**  Also closes all opened files.
      SUBROUTINE OTPUT
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN),FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)

  200 CONTINUE

  301 CALL LEGS
      CLOSE(UNIT=8,STATUS='KEEP')
      CLOSE(UNIT=30,STATUS='DELETE')

  401 CALL QDELAY
      CLOSE(UNIT=9,STATUS='KEEP')
      CLOSE(UNIT=31,STATUS='DELETE')

c  The next write is for debugging, in case row dimension in COMMON
c  blocks for matrix EVERY is exceeded.
```

```
      WRITE(UNIT=99,FMT=*) ' NUMBER OF ROWS IN EVERY = ',IEVERY
      WRITE(UNIT=99,FMT=*) ' NUMBER OF ROWS IN FLEGS = ',ILEGS
      WRITE(UNIT=99,FMT=*) ' FLIGHTS TAKEOFFS IN TIME SPAN= ',NUMFLT
      DO 210, I=1, IMSSNS
            WRITE(UNIT=99,FMT=209) I, MAXSTP(I)
  209       FORMAT('  MAXIMUM LEGS ON MISSION',I3,' =',I4)
  210 CONTINUE
      CLOSE(UNIT=99,STATUS='KEEP')

      RETURN
      END

C**   This next subroutine, called by OTPUT, provides the breakdown of
C**   what cargo is on each flight by origin-destination pairs, using
C**   the FLEGS data.  Output file created is LEGS.OUT.
      SUBROUTINE LEGS
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)
      DIMENSION DELAY(10000,11), FLEGS(9)

      CLOSE(UNIT=30,STATUS='KEEP')
      OPEN(UNIT=30,FILE='FLEGS.TMP',STATUS='OLD')
      LOWFLT = 10000
  300 READ(UNIT=30,FMT=*,END=330) (FLEGS(KK),KK=1,9)
      IF( INT(FLEGS(2)) .LT. LOWFLT) LOWFLT = INT(FLEGS(2))
      DO 320, IROW=1, ILEGS
         DO 310, N=1, ILEGS
            IF( DELAY(N,1) .EQ. 0.) THEN
               DELAY(N,1) = FLEGS(1)
               DELAY(N,2) = FLEGS(2)
               DELAY(N,3) = FLEGS(3)
               DELAY(N,4) = FLEGS(4)
               DELAY(N,5) = FLEGS(5)
               DELAY(N,6) = FLEGS(6)
               DELAY(N,7) = FLEGS(7)
               DELAY(N,8) = FLEGS(8)
               DELAY(N,9) = FLEGS(9)
               DELAY(N,10) = 1.
               DELAY(N,11) = FLEGS(9) - FLEGS(8)
               NN = NN + 1
               GOTO 300
            ELSEIF( ( DELAY(N,2) .EQ. FLEGS(2) ) .AND.
     1              ( DELAY(N,5) .EQ. FLEGS(5) ) .AND.
```

```
     2              ( DELAY(N,6) .EQ. FLEGS(6) ) .AND.
     3              ( DELAY(N,7) .EQ. FLEGS(7) ) .AND.
     4              ( DELAY(N,8) .EQ. FLEGS(8) )      ) THEN
                 DELAY(N,10) = DELAY(N,10) + 1.
                 GOTO 300
               ENDIF
   310      CONTINUE
   320   CONTINUE
   330   CONTINUE
         DO 360, IFLT=LOWFLT, LOWFLT+NUMFLT-1
            WRITE(UNIT=8,FMT=337)
   337      FORMAT(/' FLIGHT   TAIL   MISSON   LEG    CARGO  CARGO  ',
     1           'NUMBER   TIME')
            WRITE(UNIT=8,FMT=338)
   338      FORMAT( ' NUMBER  NUMBER  NUMBER  NUMBER  ORIG   DEST   ',
     1           '  OF    GOT ON'/)
            DO 350, JSTOP=1, MAXSTP(INT(FLTMSN(IFLT)))
               BOARD = 0.
               DO 340, N=1, NN
                  IF( DELAY(N,2) .EQ. REAL(IFLT)) THEN
                     CAP = DELAY(N,4)
                     IF( DELAY(N,5) .EQ. REAL(JSTOP)) THEN
                        WRITE(UNIT=8,FMT=339) INT(DELAY(N,2)),
     1                     INT(DELAY(N,3)), INT(DELAY(N,1)),
     2                     INT(DELAY(N,5)), INT(DELAY(N,6)),
     3                     INT(DELAY(N,7)), INT(DELAY(N,10)),
     4                     DELAY(N,8)
   339                  FORMAT(I5,I8,I8,I8,I7,I7,I8,F10.2)
                        BOARD = BOARD + DELAY(N,10)
                     ENDIF
                  ENDIF
   340          CONTINUE
               WRITE(UNIT=8,FMT=349) IFLT,INT(CAP),JSTOP,INT(CAP-BOARD)
   349          FORMAT(I5,5X,'CAPACITY[',I2,']',4X,I3,12X,'UNUSED: ',I2/)
   350      CONTINUE
   360   CONTINUE
         RETURN
         END

C**   This is QDELAY, called by OTPUT.  This subroutine reports how
C**   much delay cargo undergoes, by origin-destination pair.  The
C**   output file is QUEUES.OUT.
      SUBROUTINE QDELAY
      INCLUDE 'SLAM$DIR:PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
      PARAMETER (MAXQUE=15, MAXFLT=1000, MAXMSN=100)
      COMMON/UCOM1/NUMQUE, NUMFLT, NATCAR, NATFLT, ILEGS, IEVERY
      COMMON/UCOM2/ENDAT, BSTATS, ESTATS, IMSSNS, NFLOWN
      COMMON/UCOM3/MAXSTP(MAXMSN), FLTMSN(MAXFLT), CARGON(MAXFLT)
```

```
      COMMON/UCOM4/CDIRCT(MAXQUE,MAXQUE), FDIRCT(MAXMSN,13)
      COMMON/UCOM5/CMSION(MAXQUE**2,5), DEMAND(MAXQUE,MAXQUE)
      DIMENSION DELAY(10000,10), EVERY(8)

      CLOSE(UNIT=31,STATUS='KEEP')
      OPEN(UNIT=31,FILE='EVERY.TMP',STATUS='OLD')
  400 READ(UNIT=31,FMT=*,END=420) (EVERY(KK),KK=1,8)
      IF( EVERY(4) .EQ. 0.) GOTO 400
      DO 410, N=1, IEVERY
         IF( DELAY(N,1) .EQ. 0.) THEN
            DELAY(N,1) = EVERY(1)
            DELAY(N,2) = EVERY(2)
            DELAY(N,3) = EVERY(3)
            DELAY(N,4) = EVERY(4)
            DELAY(N,5) = EVERY(5)
            DELAY(N,7) = EVERY(7)
            DELAY(N,8) = EVERY(8)
            DELAY(N,9) = 1.
            DELAY(N,10) = EVERY(8)
            NN = NN + 1
            GOTO 400
         ELSEIF( ( DELAY(N,1) .EQ. EVERY(1) ) .AND.
     1           ( DELAY(N,2) .EQ. EVERY(2) ) .AND.
     2           ( DELAY(N,3) .EQ. EVERY(3) ) .AND.
     3           ( DELAY(N,4) .EQ. EVERY(4) ) ) THEN
            DELAY(N,9) = DELAY(N,9) + 1.
            DELAY(N,10) = DELAY(N,10) + EVERY(8)
            GOTO 400
         ENDIF
  410 CONTINUE
  420 CONTINUE
      WRITE(UNIT=9,FMT=427)
  427 FORMAT(/' BEGIN  END   STOP  PLANE     ON     DEPART  ',
     1        'NUMBER  MEAN     TOTAL')
      WRITE(UNIT=9,FMT=428)
  428 FORMAT( ' PORT   PORT   AT   NUMBER  FLIGHT    TIME   ',
     1        '  OF    DELAY    DELAY'/)
      DO 460, IORIG=1, NUMQUE+1
         DO 450, JDEST=1, NUMQUE
            DO 440, LPORT=1, NUMQUE
               DO 430, N=1, NN
                  IF( DELAY(N,1) .EQ. 0.) GOTO 430
                  IF( ( DELAY(N,1) .EQ. REAL(IORIG) ) .AND.
     1                ( DELAY(N,2) .EQ. REAL(JDEST) ) .AND.
     2                ( DELAY(N,3) .EQ. REAL(LPORT) ) )  THEN
                     WRITE(UNIT=9,FMT=429) INT( DELAY(N,1) ),
     1                  INT( DELAY(N,2) ), INT( DELAY(N,3) ),
     2                  INT( DELAY(N,5) ), INT( DELAY(N,4) ),
     3                  DELAY(N,7), INT( DELAY(N,9) ),
     4                  DELAY(N,10) / DELAY(N,9), DELAY(N,10)
  429                FORMAT(I4,I7,I6,I7,I8,F11.2,I6,F9.2,F9.2)
```

```
                  TEMP1 = TEMP1 + DELAY(N,10)
               ENDIF
               IF( TEMP1 .GT. 0.) THEN
                  TEMP2 = TEMP2 + TEMP1
                  TEMP1 = 0.
               ENDIF
430         CONTINUE
            IF( TEMP2 .GT. 0.) THEN
               TEMP3 = TEMP3 + TEMP2
               WRITE(UNIT=9,FMT=439) IORIG, JDEST, LPORT, TEMP2
439            FORMAT(/' SUBTOTAL DELAY FOR CARGO GOING FROM ',
   1                  I3,' TO',I3,' AT PORT',I4,' IS ',F9.2/)
               TEMP2 = 0.
            ENDIF
440       CONTINUE
          IF( TEMP3 .GT. 0.) THEN
             TEMP4 = TEMP4 + TEMP3
             WRITE(UNIT=9,FMT=447) IORIG, JDEST, TEMP3
447          FORMAT('     TOTAL DELAY FOR CARGO GOING FROM',I3,
   1               ' TO',I3,' IS ',F9.2/)
             ITEMP = ITEMP + 1
             IF( ITEMP .LT. NUMQUE ) THEN
                WRITE(UNIT=9,FMT=448)
448             FORMAT(/' BEGIN  END   STOP  PLANE     ON      ',
   1                  'DEPART  NUMBER  MEAN     TOTAL')
                WRITE(UNIT=9,FMT=449)
449             FORMAT( ' PORT   PORT   AT   NUMBER  FLIGHT    ',
   1                  ' TIME     OF    DELAY    DELAY'/)
             ENDIF
             TEMP3 = 0.
          ENDIF
450     CONTINUE
        IF( TEMP4 .GT. 0.) THEN
           TEMP5 = TEMP5 + TEMP4
           TEMP4 = 0.
        ENDIF
460  CONTINUE
     WRITE(UNIT=9,FMT=469) TEMP5
469  FORMAT(/'         TOTAL DELAY FOR ALL CARGO IS ',F9.2)
     RETURN
     END
```

Appendix F. *Description of SLAM Code and FORTRAN Inserts*

The SLAM Code. Full description of control statement (e.g., GEN, LIMITS, NETWORK) functions is beyond the scope of this research. For a detailed explanation, see (18:796–802). Other than the control statements, much of the SLAM code provided in Appendix D is repetitive in nature. These *CREATE* and *QUEUE* portions form the bulk of the code, while the remaining code is responsible for the simulation's activities.

The *CREATE* portions of the code are used for supplying the simulation with the units of cargo and aircraft which will perform activities. These *CREATE* portions also supply the attribute information about the entities which is known prior to their injection into the route system. Each *CREATE* supplies one unit of cargo or one aircraft flight. Each successive unit of cargo or flight to be supplied is done by sending a copy of the previous entity back through the *CREATE* function at the appropriate time as the simulation progresses.

The *QUEUE* portions of the code simply establish files representing the places where cargo can be located at any point in time (i.e., at an airport or in the cargo hold of a plane). With the exception of units of cargo which are DONE with their journeys, aircraft perform all actions during simulation operation and cargo is stuck in airports (a.k.a., *QUEUES*) until acted upon by a plane.

The FORTRAN inserts. The FORTRAN code for the SLAM simulation in Appendix E provides for a number of functions which the SLAM language does not directly provide. The reason for many of these FORTRAN insert functions revolves around the SLAM's inability to directly access and change the attribute values of two different type network entities at the same time. As a result, most of these functions involve the file manipulations required to pass information from aircraft boarded by cargo to the units of cargo.

Some functions allow for direct access of the information from the five data files required for simulation operation. Certain functions are designed to quickly calculate and provide information at the SLAM-level which is used for other purposes. (For instance, the calculation of the amount of time required for ground activities at a stop is determined from the aircraft's attribute information, and reported back. The SLAM code then keeps the aircraft grounded for this period of time.) Other functions provide the specialized output required for the purposes of this research.

The FORTRAN program reproduced in Appendix E is divided into the following sections or subroutines: MAIN establishes *COMMON* storage locations for information required throughout the entire SLAM and FORTRAN code and *CALL*s the SLAM processor to run the simulation; INTLC sets up the input and output files required, *READ*s the data from four input files, and inserts this data into the common variables already established by MAIN; EVENT provides the FORTRAN and SLAM processing instructions necessary for the activities shown in Figure 3.2; USERF functions allow quick calculation of information required at the SLAM level; OTPUT takes care of the final instructions needed to complete the simulation program, such as closing of opened FORTRAN files; LEGS and QDELAY provide the instructions for the specialized reports showing the cargo onboard each flight leg and the amount of cargo delay experienced, respectively.

The MAIN program actually runs the entire simulation and information established there has important consequences. The amount of computer memory allocated for SLAM processing, which is critical to simulation operation, is established by setting the "NSET" and "QSET" dimensions (See (18:389-390)). The number of flight and cargo attributes (47 and 98, respectively) is stored for subsequent use by other subroutines in FORTRAN *DO*-loop searches. Finally, variables established in MAIN (BSTATS and ESTATS) determine the specific time period within the simulation for which results are to be determined. Setting the first variable at a time other than zero (e.g., 720.) forces the simulation to progress, with cargo flowing through the

system, before delay information is obtained. Setting the second variable prior to the end of the multiple-month flight schedule (e.g., setting ESTATS equal to 1440., even though a three-month schedule will continue until 2160 hours are completed) allows the simulation to continue providing new cargo supplies as results are obtained. This circumvents problems which might occur by initially operating the simulation without cargo flowing or operating the simulation as if the supply of new cargo were suddenly stopped. These two variables, which impact other subroutines, allow the simulation to obtain information only applicable to one month's operations. The remainder of the discussion focuses on the other FORTRAN routines.

Beside establishing input and output files, INTLC *READ*s data and stores information into *COMMON* storage locations for access by other subroutines. One of these matrices of information is the cargo demand matrix. Since cargo is assumed to arrive for delivery in one-ton units, the tonnage of cargo demanded in the file is converted to a rate at which each one-ton unit arrives. This calculation divides the tonnage for each O-D pair by the number of hours in a month (i.e., 720). When used later, the demand matrix provides the number of hours between the arrivals of each one-ton unit of cargo. The OTPUT subroutine merely directs computer processing to the LEGS and QDELAY subroutines and closes files. The remainder of this explanation of the FORTRAN code focuses on the USERF functions, the EVENT routines, and finally the LEGS and QDELAY output routines.

Pritsker notes that USERF functions are particularly useful for situations where activity durations are based upon entity attributes (18:298-299). For this simulation, all times associated with aircraft flights are of this form. The time at which each unit of cargo is supplied depends on the demand for each O-D pair. To provide these times, the USERF functions perform the following actions:

1. The *CREATE* SLAM statement supplying the aircraft flight entity does not, in essence, *CREATE* flights. The *CREATE* statement supplies a dummy entity whose subsequent actions are determined through the *USERF(1)* statement.

By making a copy of this dummy entity and sending the copy back through the *CREATE* statement, subsequent calls to the USERF function are made, until the USERF function determines that the supply of flights has been exhausted. Thus, the supply of flights really comes from the first USERF function. The SLAM code only really needs to know how long to hold the dummy entity before allowing it to proceed to the next activities (i.e., departing on the first leg and going back through the CREATE sequence). This *USERF(1)* function accomplishes the following actions:

- First, a flight's attributes are established by *READ*ing a set of data values stored in the flight schedule file (See Appendix Q). The attribute values correspond directly with the fields of information provided in this file, created by the FORTRAN scheduling programs.
- Second, a count is maintained of the number of flights whose attributes have been *READ* (a.k.a., the number of flights departing).
- Finally, the function calculates the time between the current simulation time and the time when the flight needs to depart (i.e., field 9). The time calculated is reported to the SLAM code, which will not do anything with the entity(ies) until this amount of time has elapsed.

2. The *USERF(2)* function looks though an aircraft's attributes for the time required to fly to the next location (i.e., fields 10, 13, ..., 46) based on where the aircraft is currently stopped (i.e., field 7).

3. The *USERF(3)* function simply determines the numerical representation of the airport where the aircraft currently is located (i.e., fields 8, 11, ... 47), based on the current stop number (i.e., field 7).

1. The *USERF(4)* function looks through an aircraft's attributes for the ground time (i.e., fields 9, 12, ..., 46) applicable to the current stop number (i.e., field 7).

5. All the *cargo CREATE* portions of the SLAM code use the last function, *USERF(5)*, to calculate when to supply the next unit of cargo of the applicable type. When these times are requested, SLAM has direct access to the attributes of a cargo unit; therefore, this is the only USERF function not based on aircraft attribute values. This function is simply a table look-up of information stored in the demand matrix, based on which O-D pair of cargo (i.e., fields 2 and 3) is involved.

Before discussing the FORTRAN *EVENT*s, an explanation of the *TEMP QUEUE* alleviates the need for separate discussion in each *EVENT* description which follows. Because the *HOLD QUEUE* stores all cargo onboard all aircraft and each airport *QUEUE* stores all cargo at a particular location, computer searches through these files might become cumbersome. For this reason, a *TEMP QUEUE* has been utilized to limit the number of computer search operations which might be required. For each search through a file (a.k.a., *QUEUE*), each unit of cargo is removed from the current file location and examined. When the unit of cargo fulfills certain criteria, the attributes of the cargo might be changed. When the criteria are not met, the cargo attributes usually remain intact. In either case, the cargo is either filed temporarily in the *TEMP QUEUE* or refiled in a *QUEUE* other than the one currently being searched through. This forces the computer to examine the units of cargo in the file one at a time, until all cargo in the *HOLD* or airport *QUEUE* has been examined. After all units of cargo have been examined, all units of cargo filed in the *TEMP QUEUE* are removed and refiled back in the original file searched.

Also, to eliminate confusion of the method used for comparing the attribute values of aircraft and cargo, understanding of certain SLAM peculiarities may be helpful. SLAM can normally only access the attribute values of one entity at a time, through the "ATRIB" array. For this research, the attribute values available in this "ATRIB" array usually belong to an aircraft flight entities. SLAM provides subprograms, useable at the FORTRAN-insert level, which can access and change attribute

values of an entity other than the entity currently being processed through a SLAM statement. Much like this research's use of the temporary *QUEUE* just discussed, SLAM can temporarily copy an entity's attributes into another array, similar to the "ATRIB" array. The second entity's attributes can thus be accessed and changed, when required, by placement of the values into the other array, and storing the other array's values into files when required. SLAM already has predefined subprograms which perform these functions. "Subroutine RMOVE(NRANK, IFILE,A) removes an entry with rank NRANK from file IFILE and places its attributes into the buffer array A" (18:297). "Subroutine FILEM(IFILE,A) files an entry with attributes specified in the buffer array A into file IFILE)" (18:297). The computer instructions through which this research makes comparisons between flight and cargo attributes relies heavily on SLAM subprograms like these.

The last important aspect relating to activities occurring during a flight's processing through the *EVENT*s relates to *time*. Simulation clock time stands still as each *EVENT* is processed. The only activities which take *time* are flying and sitting on the ground. With these general aspects in mind, a discussion of each *EVENT* follows.

During the *FILL EVENT*, the *HOLD* is searched for all cargo identified as currently residing on the aircraft which is about to depart on a flight leg. For each unit of cargo found onboard the aircraft, a search of the cargo entity's attributes determines the next set of attribute fields which have not yet been filled. These attribute fields are now filled in, telling the cargo that their journey is about to continue. Figure F.1 shows how these searches and file manipulations are accomplished.

The simulation does not implicitly know where each plane is located at every instant in time. For this reason several *EVENT*s, including *PNEW* first determine the airport where the aircraft is currently located. This *EVENT* is specifically designed to *Pickup NEW* cargo at the current airport, when aircraft space allows. To pickup cargo, a search looks through all cargo currently located at the airport for

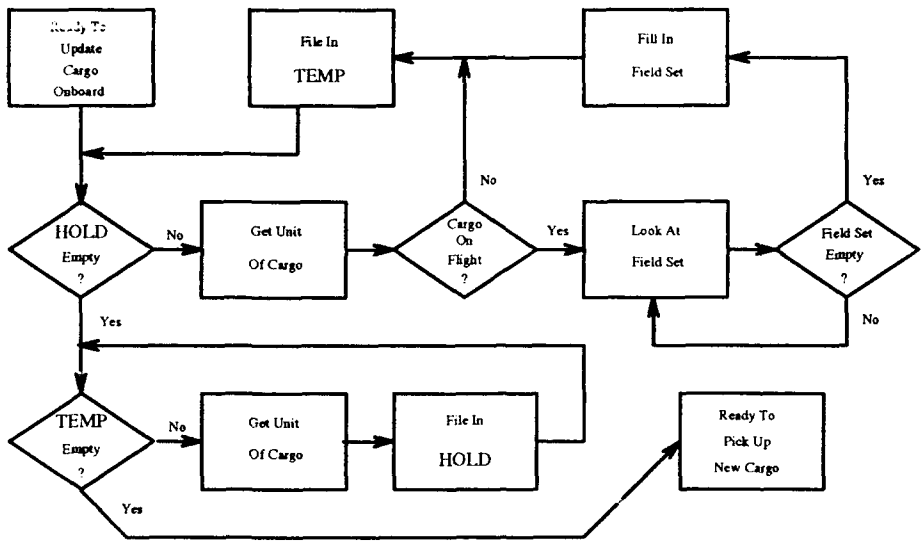


Figure F.1. The FILL EVENT

cargo for which the decision rules indicate pickup is appropriate. A unit of cargo must need to travel in the same direction as the aircraft. The cargo must either not require a specific type of mission or require the type of mission to which the flight is assigned. If these criteria are met, the cargo is *placed* onboard the aircraft (i.e., filed in the *HOLD*), after the cargo's attributes are updated with the information needed to document the next segment of the cargo's journey. If the criteria are not met, the unit of cargo remains at the airport until another aircraft can pick it up. No other *EVENT*s assign cargo to aircraft.

Just before arrival at a stop, the SLAM *ASSIGN* statement increments the stop number (i.e., field 7) of the aircraft flight entity. Without this statement, the aircraft (and subsequently the cargo onboard) would not know of arrival at the next airport along the route. Thus when the computer processes a flight through the *DROP EVENT*, the *flying* activity is done and determination must be made whether to *drop* cargo at this new airport. Each unit of cargo is removed from the *HOLD* to find the cargo on the specific plane. For all cargo on the aircraft, the origin, destination, and time at which the cargo first *got on* the aircraft are stored in

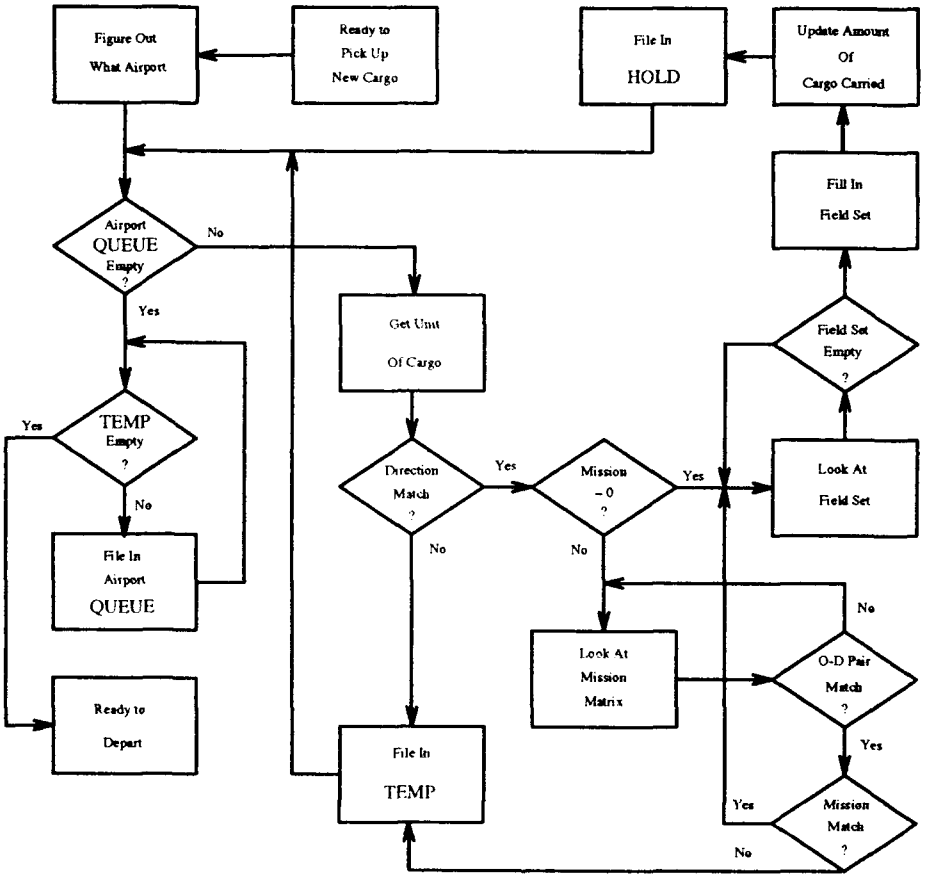


Figure F.2. The PNEW EVENT

a temporary working file, as long as the flight's arrival falls within the information gathering window (i.e., between BSTATS and ESTATS discussed previously). This provides a manifest listing of all cargo on the aircraft during the flight leg just flown. The computer updates each unit of cargo's current location attribute (i.e., field 7), informing the cargo of arrival at the new location. When the current location matches the cargo destination, the cargo's journey is completed, so the cargo is now (filed in) *DONE*. Otherwise the cargo remains on the aircraft until further offload criteria can be applied — remember that no time elapses between the beginning of the *DROP* and the conclusion of the *FFIN EVENT*s. The decision logic associated with the *PNEW EVENT* is shown in Figure F.2 and the *DROP EVENT* is shown in Figure F.3.

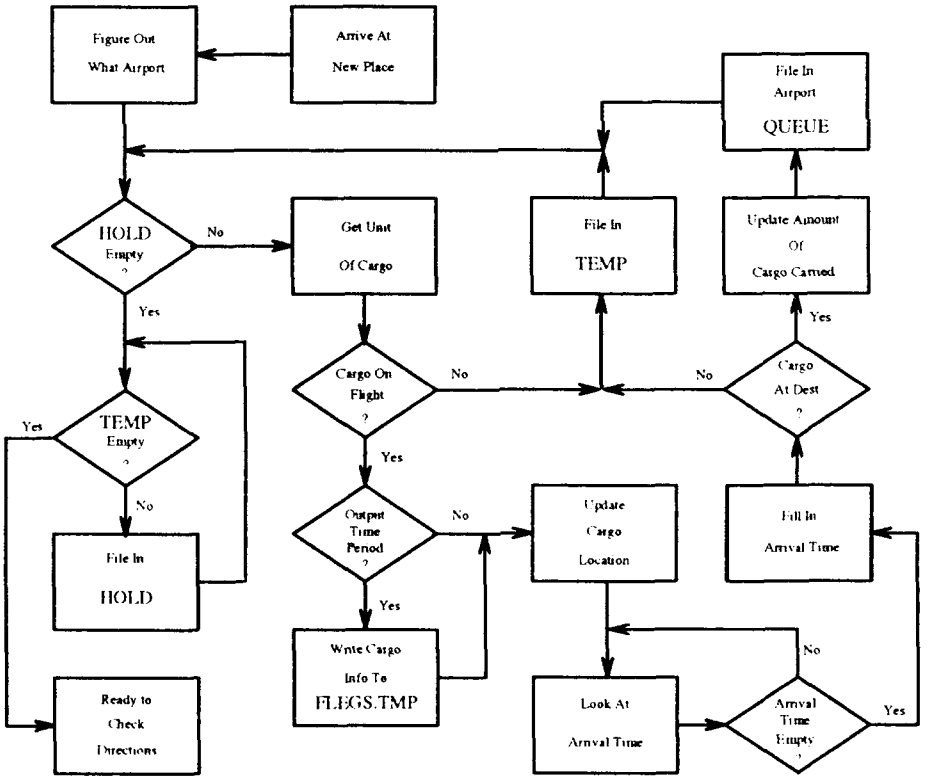


Figure F.3. The DROP EVENT

Before decisions can be made whether cargo should remain onboard an aircraft, the aircraft direction, as well as the cargo direction and mission needs, must reflect the current location. The *FDIR* and *CDIR EVENT*s provide these updates to flight and cargo attributes. The *FDIR EVENT* simply supplies the direction of a flight's next leg. As the operations occurring in CDIR are more complex, the activity flow is shown in Figure F.4. As can be seen from Figure F.4, the entities processed through this *EVENT* are aircraft, even though only cargo attributes change. Cargo found onboard the particular flight have their mission-to-get-on-flag and direction attributes updated, based on the current location (previously adjusted during the *DROP EVENT*). Both the *FDIR* and *CDIR EVENT*s access information *READ* in during the *INTLC* initial subroutine.

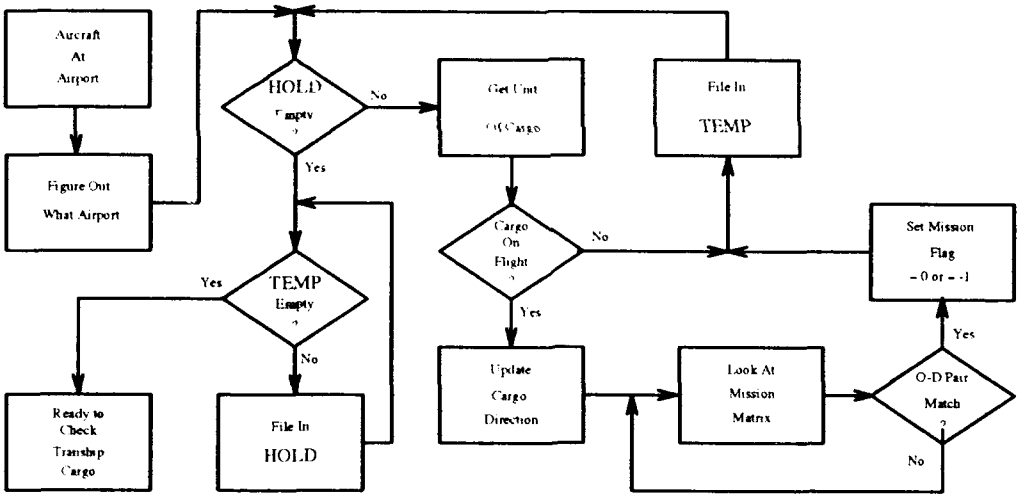


Figure F.4. The CDIR EVENT

The *DOFF EVENT*, shown in Figure F.5, decides whether cargo should remain onboard an aircraft which has arrived at a new location. This entire *EVENT* is skipped when an aircraft's current location is the same as the home base — the flight is finished. After determining the current location from the aircraft flight attributes, examination of cargo onboard all aircraft (a.k.a., filed in the *HOLD*) pulls out all units on the specific flight. For these units of cargo to stay on the flight, the flight's

direction must match the cargo direction; and the cargo must either not require a specific mission or the flight's mission must be one which the cargo needs for travel further along its journey. When this criteria fails, the unit of cargo is *dropped* at the location (a.k.a., filed in the appropriate airport *QUEUE*) to await transshipment on another flight.

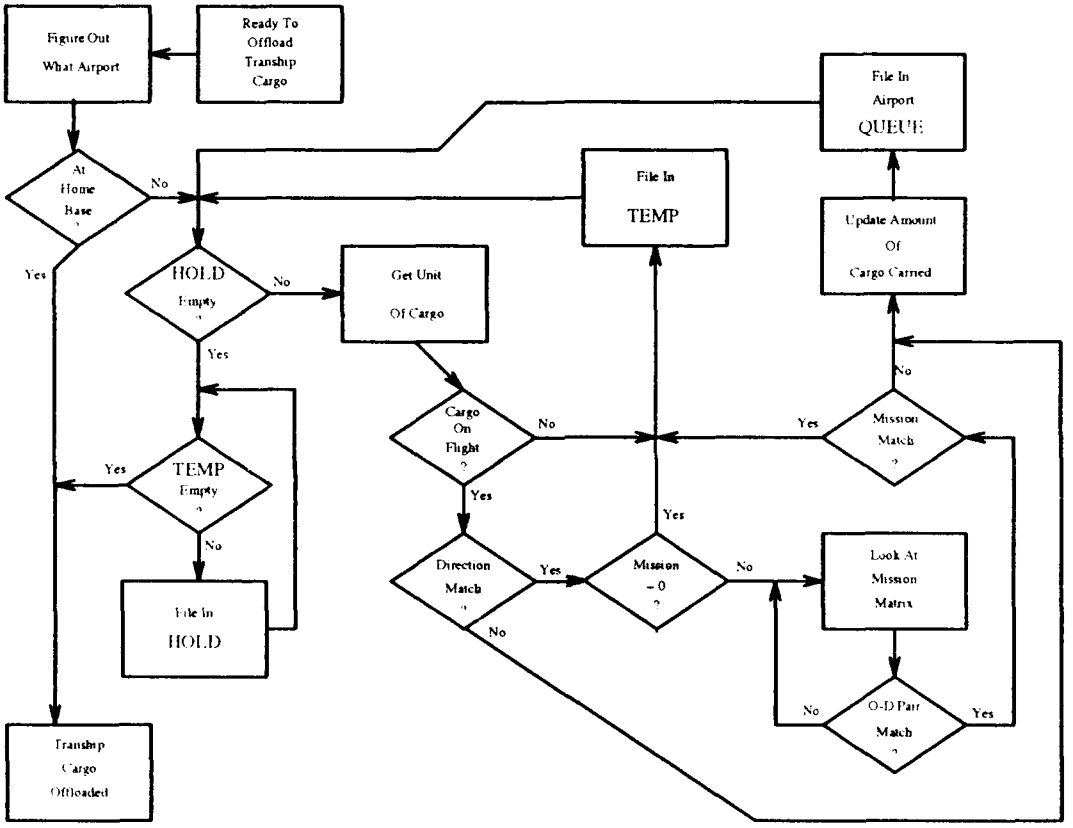


Figure F.5. The DOFF EVENT

Every flight eventually completing its mission and returning to its home base might have cargo onboard. Since the flight terminates at the home base airport, all cargo carried must be offloaded to await another flight. The *FFIN EVENT*, depicted in Figure F.6, handles these situations. The computer skips this *EVENT* when the airport location is not the home base. Each unit of cargo found onboard (a.k.a., filed

in the *HOLD*) the specific aircraft are ***offloaded*** at the airport (a.k.a., refiled in the airport *QUEUE*).

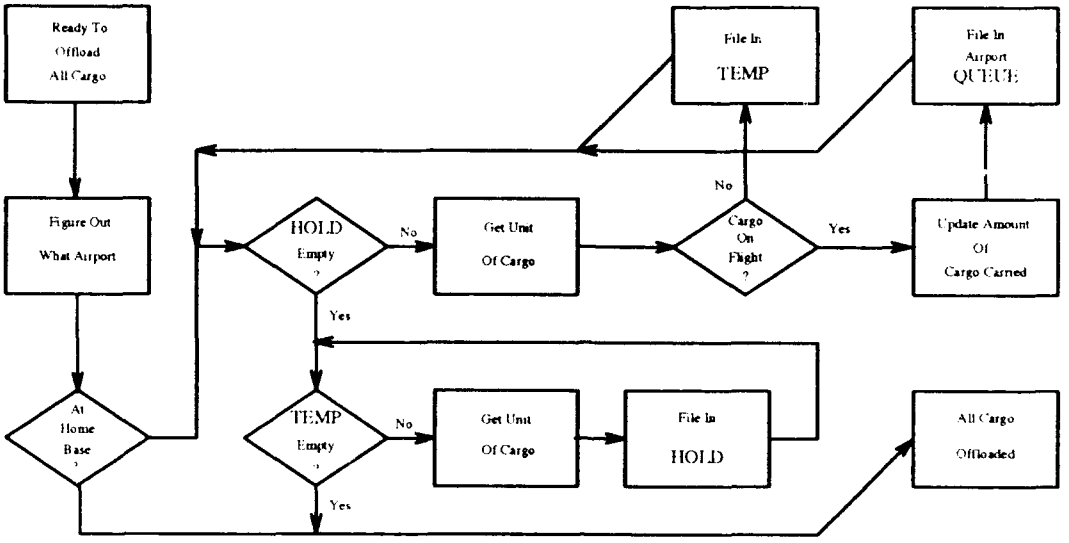


Figure F.6. The FFIN EVENT

The previous *EVENT*s (along with the SLAM *SIT* and *FLY* activities) describe the entire sequence of events applicable to processing aircraft flight entities through the simulation. One more *EVENT* affects cargo entities prior to their elimination from the network. This *CARGo EVENT* stores the information about every unit of cargo's journey into a temporary working file. For each segment of a cargo entity's journey, a record is kept of the cargo's origin, the cargo's destination, the number of the flight taken on the segment, the tail number taken on the segment, the location where stopped, the time departed the last stop, the time departed the current stop, and the time elapsed (a.k.a., the delay experienced) on the segment. This file will only contain information applicable to the time period within which results are to be obtained (i.e., within BSTATS and ESTATS). Units of cargo completing their journeys in the applicable information-gathering window may have traveled on flights prior to that time.

The output subroutine *LEGS* looks through the temporary file for aircraft and consolidates the information applicable to each individual flight leg. The consolidation groups all cargo entities (on each specific flight leg) having the same origin, destination, and time put onboard. The subroutine thus provides information about how much of the aircraft's capacity has been utilized on each flight and what specific cargo were onboard. For each cargo group the output shows the total number of cargo units which share the same origin, destination, and time onboard. The output provides a synopsis, by flight, of the cargo groups onboard. These results are captured in an output file, an excerpt of which is provided in Appendix S.

The output subroutine *QDELAY* groups information in a manner similar to the *LEGS* subroutine. *QDELAY* uses the information stored in the temporary file for cargo histories to determine the delay experienced by all cargo entities within the time period for obtaining results. The smallest group depicted are those units of cargo of the same O-D pair, on the same flight, at the same stop location, which have experienced the same amount of delay. The *QDELAY* subroutine subtotals the delay experienced at each airport and totals the overall delay experience for each O-D cargo pair. The results obtained from this subroutine are stored in another output file. Appendix R is an excerpt of an example file.

Appendix G. *The BASE.INP File*

This file provides a listing of all airports encompassing the route system – all airports may not be visited in any given month. A computer associates a number with each of the International Civil Aviation Organization (ICAO) airport codes, in the order listed in the file. Either numerical or ICAO representations can be used to refer to specific airports. In the example below, airport *1* and airport *AAAA* would be synonymous. The ICAO codes herein were made up for the hypothetical twelve-airport route system. This input file is required for MAC's FORTRAN scheduling program found in Appendix A. The file follows:

```
AAAA
BBBB
CCCC
DDDD
EEEE
FFFF
GGGG
HHHH
IIII
JJJJ
KKKK
LLLL
MMMM
NNNN
OOOO
PPPP
```

Appendix H. *The ROUTE.INP File*

This file describes all routes which will be used for the month, as would be determined by the MAC computer channel route model's linear programming relaxation. Each route is listed on a separate line with airport ICAO codes and numerical reason for stopping. The order in which routes are listed in the file implicitly associates a route with a mission number. The routes depicted herein were used for the hypothetical twelve-airport route system. This input file is required for MAC's schedule-creation program found in Appendix A. The file follows:

```
AAAA1 CCCC6 IIII6 EEEE6 AAAA9
AAAA1 DDDD6 KKKK6 DDDD6 AAAA9
AAAA1 FFFF6 AAAA9
BBBB1 AAAA4 BBBB9
BBBB1 AAAA4 FFFF6 BBBB9
BBBB1 CCCC6 BBBB9
BBBB1 DDDD4 FFFF6 EEEE4 BBBB9
IIII1 GGGG4 EEEE6 DDDD4 CCCC6 IIII9
IIII1 HHHH4 JJJJ4 IIII9
IIII1 JJJJ4 HHHH4 IIII9
IIII1 KKKK4 FFFF6 DDDD4 GGGG6 IIII9
KKKK1 FFFF6 CCCC4 DDDD6 EEEE4 KKKK9
KKKK1 IIII4 GGGG4 HHHH6 JJJJ4 KKKK9
KKKK1 LLLL6 FFFF6 IIII4 KKKK9
```

Appendix I. *The GNDTM.INP File*

This file tells the number of flights required for each mission and the type of aircraft used for these flights. The MAC computer channel route model's linear programming relaxation normally provides this information. The first figure on each line is the number of flights, and the second figure is the aircraft type. In the example below, eight flights of mission one would be required, and these missions would be flown using C-5 (i.e., type 1) aircraft. The figures appearing herein were for the hypothetical twelve-airport route system. This input file is required for the two schedule-creating programs found in Appendices A and B. The file follows:

```
 8      1
 4      1
24      1
 6      2
43      2
20      2
 8      2
10      3
10      3
10      3
10      3
12      2
12      2
12      2
```

Appendix J. *The GNDTM.INP File*

This file gives the amounts of time required by the various ground activities for which a plane stops and the aircraft cargo capacities, by aircraft type. The columns, except the last, coincide with numerical codes designating the reason for stopping: the first is for mission commencement, explaining why zeroes are found in the entire column; the next two are for onload and offload times, respectively; the fourth and fifth are the time for enroute fuel and aircrew change; enroute crew rest time is in the sixth column; the seventh and eighth columns are not currently used. The last column provides the number of one-ton cargo payloads which a type of aircraft can hold. The lines of data provide the information applicable to C-5, C-141, C-130, DC-8, DC-10, B-747, and C-17 aircraft — in that order. Programs refer to aircraft type by the order listed in this file. Thus, a C-5 is aircraft type *1*, a C-141 is aircraft *2*, and so on. The figures appearing were provided by HQ MAC/XPYR and were used intact with the hypothetical twelve-airport route system. This input file is required for the two FORTRAN schedule-creating programs found in Appendices A and B. The file follows:

```
0   4.25   4.25   4.25   4.25   18.25   4.25   4.25   50
0   3.25   3.25   3.25   3.25   17.25   3.25   3.25   20
0   2.25   2.25   2.25   2.25   16.25   2.25   2.25    7
0   3.00   3.00   3.00   3.00   16.00   3.00   3.00   25
0   4.00   4.00   4.00   4.00   16.00   4.00   4.00   40
0   4.00   4.00   4.00   4.00   16.00   4.00   4.00   71
0   3.25   3.25   3.25   3.25   17.25   3.25   3.25   32
```

Appendix K. *The FLY.DAT File*

The flying times between locations are in this file. Each row of data includes a takeoff location, a landing location, and the time to fly between the points. HQ MAC/XPYR uses flying times based on historical experience. For the hypothetical twelve-airport route system the times shown below were used. These times approximate the times used by HQ MAC/XPYR. This input file is required for the two FORTRAN schedule-creation programs found in Appendices A and B. The file follows:

```
1    2    1.10
1    3    7.90
1    4    7.80
1    6    7.90
2    1    1.10
2    3    8.20
2    4    7.50
2    5    7.70
3    2    9.70
3    1    9.40
3    5    1.40
3    4    1.60
3    6    1.40
3    9    6.20
4    1    8.70
4    3    1.80
4    5    0.70
4    6    2.10
4    7    2.20
4    9    3.30
4   11    4.00
5    1    9.10
5    2    8.70
5    3    1.70
5    4    1.20
5    6    2.50
5    9    3.10
5   11    4.20
6    1    9.90
6    2   10.40
6    3    1.50
6    4    1.80
6    5    2.50
```

6	7	2.00
6	9	3.10
6	11	4.40
6	12	7.30
7	5	2.30
7	6	2.20
7	8	1.50
7	9	1.50
8	9	0.50
8	10	0.90
9	4	3.60
9	5	3.70
9	6	3.50
9	7	1.70
9	8	0.60
9	10	0.50
9	11	2.10
10	8	1.00
10	9	0.80
10	11	1.10
11	4	4.30
11	6	5.20
11	9	2.50
11	12	3.10
12	6	8.80
12	11	1.10

Appendix L. *The ROUTE.DAT File*

As this file provides the foundation upon which the detailed *flight* schedule is based, this file deserves special description.

The number of lines included for each mission depends upon how many tail numbers will be assigned for flying the number of flights for each mission type, along with the number of airport locations where these aircraft stop. Each line of information breaks down as follows: a tail number, a stop number, the airport number stopped at, a time, and the mission number to which the tail number is assigned.

The number of lines for each tail number ties directly to the number of places stopped at for the mission type, as defined in the route file. For the route *AAAA1 BBBB6 AAAA9*, each tail number flying this route will have three lines in this file — one for stopping at (departing from) airport 1, one for stopping (resting) at airport 2, and one for stopping (ending up) at airport 1.

The times listed in the file serve two purposes. The time listed on a tail number's first line provides the time during the month (i.e., a 720-hour period of time) when that tail number departs on its first flight. (After establishing the first takeoff, this time becomes irrelevant.) The times listed for a tail number's subsequent lines provide the amount of time the aircraft will remain on the ground until taking off again. Of particular note is the time listed on a tail number's last line (i.e., mission stopping back at the home base). This time is the ground time until the next flight for that particular tail number commences.

As defined in this schedule, each tail number would fly the route repeatedly, as long as a simulation program would allow time to proceed.

The information shown in the file below were used for the hypothetical twelve-airport route system. This type of file is directly output from MAC's schedule-

creation program found in Appendix A. This file also serves as input for the flight schedule-creation program found in Appendix B. The file follows:

```
 1    1    1     0.00    1
 1    2    3    18.25    1
 1    3    9    18.25    1
 1    4    5    18.25    1
 1    5    1     8.35    1
 2    1    1     2.50    2
 2    2    4    18.25    2
 2    3   11    18.25    2
 2    4    4    18.25    2
 2    5    1   100.45    2
 3    1    1     5.00    3
 3    2    6    18.25    3
 3    3    1    23.95    3
 4    1    1    35.00    3
 4    2    6    18.25    3
 4    3    1    23.95    3
 5    1    2     7.50    4
 5    2    1     3.25    4
 5    3    2   114.55    4
 6    1    2    10.00    5
 6    2    1     3.25    5
 6    3    6    17.25    5
 6    4    2    10.10    5
 7    1    2    26.67    5
 7    2    1     3.25    5
 7    3    6    17.25    5
 7    4    2    10.10    5
 8    1    2    43.33    5
 8    2    1     3.25    5
 8    3    6    17.25    5
 8    4    2    10.10    5
 9    1    2    12.50    6
 9    2    3    17.25    6
 9    3    2     0.85    6
10    1    2    15.00    7
10    2    4     3.25    7
10    3    6    17.25    7
10    4    5     3.25    7
10    5    2    45.45    7
11    1    9    17.50    8
11    2    7     2.25    8
11    3    5    16.25    8
11    4    4     2.25    8
11    5    3    16.25    8
11    6    9    21.80    8
12    1    9    20.00    9
12    2    8     2.25    9
```

12	3	10	2.25	9
12	4	9	65.20	9
13	1	9	22.50	10
13	2	10	2.25	10
13	3	8	2.25	10
13	4	9	65.50	10
14	1	9	25.00	11
14	2	11	2.25	11
14	3	6	16.25	11
14	4	4	2.25	11
14	5	7	16.25	11
14	6	9	22.20	11
15	1	11	27.50	12
15	2	6	17.25	12
15	3	3	3.25	12
15	4	4	17.25	12
15	5	5	3.25	12
15	6	11	5.80	12
16	1	11	30.00	13
16	2	9	3.25	13
16	3	7	3.25	13
16	4	8	17.25	13
16	5	10	3.25	13
16	6	11	25.30	13
17	1	11	32.50	14
17	2	12	17.25	14
17	3	6	17.25	14
17	4	9	3.25	14
17	5	11	5.15	14

Appendix M. *The FDIRCT.DAT File*

This file provides directional guidance associated with each flight leg. Each line of the file provides directional guidance for one route. The first number is the direction of travel on the first flight leg; the second number is direction on the second leg, and so on. Twelve flight legs are allowed per route.

The directional codes in this research were "5," "10," and "15." The "5" represents travel in one direction, while the "15" represents travel in the opposite direction. This could be thought of as representing east and west, for instance. The "10" represents a neutral direction, which allows the placement of cargo on aircraft going in *either* direction. The program user must establish directional codes which adequately describe the route system in use.

The figures appearing herein were used for the hypothetical twelve-airport route system. This input file is required for the schedule-creation program found in Appendix C and for the SLAM FORTRAN insert program found in Appendix E. The file follows:

```
 5. 5.15.15.00.00.00.00.00.00.00.00.
 5. 5.15.15.00.00.00.00.00.00.00.00.
 5.15.00.00.00.00.00.00.00.00.00.00.
10.15.00.00.00.00.00.00.00.00.00.00.
 5. 5.15.00.00.00.00.00.00.00.00.00.
 5.15.00.00.00.00.00.00.00.00.00.00.
 5. 5.15.15.00.00.00.00.00.00.00.00.
15.15.15.15. 5.00.00.00.00.00.00.00.
15. 5.15.00.00.00.00.00.00.00.00.00.
 5.15. 5.00.00.00.00.00.00.00.00.00.
 5.15.15. 5. 5.00.00.00.00.00.00.00.
15.15. 5. 5. 5.00.00.00.00.00.00.00.
15.15. 5. 5. 5.00.00.00.00.00.00.00.
 5.15. 5. 5.00.00.00.00.00.00.00.00.
```

Appendix N. *The CDIRCT.DAT File*

This file provides directional guidance to cargo, based on the cargo's current location and intended destination. Each line of the file provides direction for one *location-destination* pair — not to be confused with O-D pairs. The first number is the airport number where the cargo is currently located. The second number is the number of the destination airport. The final number is the direction which the cargo needs to travel.

The directional codes currently used in this research were "5," "10," and "15." The "5" represents travel in one direction, while the "15" represents travel in the opposite direction. This could be thought of as representing east and west, for instance. The "10" represents a neutral direction, which allows the placement of cargo on aircraft going in *either* direction. The program user must establish directional codes which adequately describe the route system in use.

The figures appearing herein were used for the hypothetical twelve-airport route system. This input file is required for the SLAM FORTRAN insert program found in Appendix E. The file follows:

```
1    2   15.
1    3    5.
1    4    5.
1    5    5.
1    6    5.
1    7    5.
1    8    5.
1    9    5.
1   10    5.
1   11    5.
1   12    5.
2    1    5.
2    3    5.
2    4    5.
2    5    5.
2    6    5.
2    7    5.
2    8    5.
```

```
2   9    5.
2  10    5.
2  11    5.
2  12    5.
3   1   15.
3   2   15.
3   4    5.
3   5    5.
3   6    5.
3   7    5.
3   8    5.
3   9    5.
3  10    5.
3  11    5.
3  12    5.
4   1   15.
4   2   15.
4   3   10.
4   5    5.
4   6    5.
4   7    5.
4   8    5.
4   9    5.
4  10    5.
4  11    5.
4  12    5.
5   1   15.
5   2   15.
5   3   15.
5   4   15.
5   6    5.
5   7    5.
5   8    5.
5   9    5.
5  10    5.
5  11    5.
5  12    5.
6   1   15.
6   2   15.
6   3   15.
6   4   15.
6   5   15.
6   7   10.
6   8   10.
6   9   10.
6  10   10.
6  11    5.
6  12    5.
7   1   10.
7   2   10.
7   3   15.
```

```
 7   4   15.
 7   5   10.
 7   6   10.
 7   8    5.
 7   9    5.
 7  10    5.
 7  11    5.
 7  12    5.
 8   1    5.
 8   2    5.
 8   3    5.
 8   4    5.
 8   5    5.
 8   6    5.
 8   7    5.
 8   9    5.
 8  10    5.
 8  11    5.
 8  12    5.
 9   1   10.
 9   2   10.
 9   3   10.
 9   4   10.
 9   5   15.
 9   6   10.
 9   7   15.
 9   8   10.
 9  10   10.
 9  11   10.
 9  12   10.
10   1   10.
10   2   10.
10   3   10.
10   4   10.
10   5   10.
10   6   10.
10   7   10.
10   8   10.
10   9   10.
10  11    5.
10  12    5.
11   1   15.
11   2   15.
11   3   15.
11   4   15.
11   5   15.
11   6   15.
11   7   15.
11   8   15.
11   9   15.
11  10   15.
```

```
11  12    5.
12   1   15.
12   2   15.
12   3   15.
12   4   15.
12   5   15.
12   6   15.
12   7   15.
12   8   15.
12   9   15.
12  10   15.
12  11   15.
```

Appendix O. *The CMSSION.DAT File*

This file supplies the user-provided decision criteria defining which, if any, specific missions cargo must fly on. The first two numbers on a line are the numerical representations for the *location-destination* pair. The other three numbers specify which missions cargo must be on, or wait for, from the current location to reach the destination. When all three mission numbers are zero, the cargo is free to get on any available mission going in the correct direction.

The figures appearing herein were used for the hypothetical twelve-airport route system. For another system, the user would need to determine whether cargo must travel by certain missions to get to their destination. This input file is required for the SLAM FORTRAN insert program found in Appendix E. The file follows:

```
1   2   4   0   0
1   3   1   0   0
1   4   2   0   0
1   5   1   2   3
1   6   3   5   0
1   7   0   0   0
1   8   0   0   0
1   9   1   2   0
1  10   0   0   0
1  11   0   0   0
1  12   1   2   0
2   1   4   5   0
2   3   4   6   0
2   4   4   5   7
2   5   4   7   0
2   6   4   7   0
2   7   4   6   7
2   8   4   6   7
2   9   4   6   7
2  10   4   6   7
2  11   4   6   7
2  12   4   6   7
3   1   0   0   0
3   2   6   0   0
3   4  12   0   0
3   5   1  12   0
3   6   0   0   0
```

3	7	0	0	0
3	8	0	0	0
3	9	1	0	0
3	10	0	0	0
3	11	0	0	0
3	12	0	0	0
4	1	2	0	0
4	2	0	0	0
4	3	2	7	8
4	5	12	0	0
4	6	7	0	0
4	7	0	0	0
4	8	0	0	0
4	9	0	0	0
4	10	0	0	0
4	11	0	0	0
4	12	0	0	0
5	1	1	7	0
5	2	1	7	0
5	3	0	0	0
5	4	0	0	0
5	6	0	0	0
5	7	0	0	0
5	8	0	0	0
5	9	0	0	0
5	10	0	0	0
5	11	0	0	0
5	12	0	0	0
6	1	3	5	0
6	2	0	0	0
6	3	12	0	0
6	4	0	0	0
6	5	7	11	12
6	7	11	14	0
6	8	11	14	0
6	9	14	11	0
6	10	14	11	0
6	11	0	0	0
6	12	0	0	0
7	1	8	11	0
7	2	8	11	0
7	3	0	0	0
7	4	0	0	0
7	5	8	11	0
7	6	0	0	0
7	8	0	0	0
7	9	0	0	0
7	10	0	0	0
7	11	0	0	0
7	12	0	0	0
8	1	0	0	0

8	2	0	0	0
8	3	0	0	0
8	4	0	0	0
8	5	0	0	0
8	6	0	0	0
8	7	0	0	0
8	9	0	0	0
8	10	0	0	0
8	11	0	0	0
8	12	0	0	0
9	1	1	14	0
9	2	1	14	0
9	3	1	14	0
9	4	1	8	11
9	5	1	8	0
9	6	11	14	0
9	7	8	13	0
9	8	9	10	13
9	10	9	10	13
9	11	11	13	14
9	12	11	13	14
10	1	0	0	0
10	2	0	0	0
10	3	0	0	0
10	4	0	0	0
10	5	0	0	0
10	6	0	0	0
10	7	0	0	0
10	8	9	10	0
10	9	9	10	0
10	11	0	0	0
10	12	0	0	0
11	1	0	0	0
11	2	0	0	0
11	3	2	12	0
11	4	2	0	0
11	5	0	0	0
11	6	11	12	0
11	7	13	0	0
11	8	13	0	0
11	9	13	0	0
11	10	13	0	0
11	12	0	0	0
12	1	0	0	0
12	2	0	0	0
12	3	0	0	0
12	4	0	0	0
12	5	0	0	0
12	6	0	0	0
12	7	0	0	0
12	8	0	0	0

```
12   9   0   0   0
12  10   0   0   0
12  11   0   0   0
```

Appendix P. *The DEMAND.DAT File*

This file supplies the tonnage of cargo demand for O-D pairs. In each line of data, the first two numbers are numerical representations of the origin and destination airports. The third number is the quantity, in tons, of cargo for the origin-destination pair. The figures appearing herein were used for the hypothetical twelve-airport route system. This input file is required for the cargo simulation insert program found in Appendix E. The file follows:

```
1   4    73.13
1   6  2067.40
1   9   183.37
1  12   249.98
2   3   300.00
2   5   111.60
2   7   130.27
2   8   222.52
2  10    40.27
2  12   121.50
3   2   209.25
3   4     4.28
3   6    42.98
3  10     5.63
4   1    46.35
4   6     9.45
4   9     9.22
4  10     5.63
5   2    71.78
5   4     9.68
5   7    17.55
5   8    32.40
5  12    22.50
6   1  1160.55
6   3    62.10
6   4    27.23
6   9   183.37
6  10    10.80
7   2    47.47
7   5    28.35
7   8    32.40
8   2   144.90
8   4     8.55
8   5    49.95
8   7    35.10
```

9	1	61.20
9	4	11.70
9	6	20.92
10	1	29.47
10	3	8.10
10	4	9.45
10	6	4.05
12	1	240.00
12	2	52.42
12	5	14.63

Appendix Q. *Excerpt of SRAW1.DAT File*

This file is the output created by the FORTRAN flight scheduling program found in Appendix B. This file is also the input file used by the FORTRAN program found in Appendix C. The format of this file is exactly the same as the file created by the FORTRAN program found in Appendix C, which is the *SCHED.DAT* file. The *SCHED.DAT* file is the input file used to schedule flights for the SLAM simulation found in Appendices D and E. An excerpt of the file created for the hypothetical twelve-airport route system follows:

```
 1.  1. 1.50. 5. 1. 0. 1.   0.00 7.90 3.
18.25 6.20 9.18.25 3.70 5.18.25 9.10 1. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 2.  2. 2.50. 5. 1. 0. 1.   2.50 7.80 4.
18.25 4.0011.18.25 4.30 4.18.25 8.70 1. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 3.  3. 3.50. 5. 1. 0. 1.   5.00 7.90 6.
18.25 9.90 1. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 4.  4. 5.20.10. 2. 0. 2.   7.50 1.10 1.
 3.25 1.10 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 5.  5. 6.20. 5. 2. 0. 2.  10.00 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 6.  6. 9.20. 5. 2. 0. 2.  12.50 8.20 3.
17.25 9.70 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 7.  7.10.20. 5. 2. 0. 2.  15.00 7.50 4.
 3.25 2.10 6.17.25 2.50 5. 3.25 8.70 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 8.  8.11. 7.15. 9. 0. 9.  17.50 1.70 7.
 2.25 2.30 5.16.25 1.20 4. 2.25 1.80 3.16.25 6.20 9. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0  0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 9.  9.12. 7.15. 9. 0. 9.  20.00 0.60 8.
 2.25 0.9010. 2.25 0.80 9. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
10. 10.13. 7. 5. 9. 0. 9.  22.50 0.5010.
 2.25 1.00 8. 2.25 0.50 9. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
11. 11.14. 7. 5. 9. 0. 9.  25.00 2.1011.
 2.25 5.20 6.16.25 1.80 4. 2.25 2.20 7.16.25 1.50 9. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
```

```
 5. 12. 7.20. 5. 2. 0. 2.  26.67 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
12. 13.15.20.15.11. 0.11.  27.50 5.20 6.
17.25 1.50 3. 3.25 1.60 4.17.25 0.70 5. 3.25 4.2011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
13. 14.16.20.15.11. 0.11.  30.00 2.50 9.
 3.25 1.70 7. 3.25 1.50 8.17.25 0.9010. 3.25 1.1011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
14. 15.17.20. 5.11. 0.11.  32.50 3.1012.
17.25 8.80 6.17.25 3.10 9. 3.25 2.1011. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
  .
  .
  .
 6.100. 9.20. 5. 2. 0. 2. 372.50 8.20 3.
17.25 9.70 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 7.101.10.20. 5. 2. 0. 2. 375.00 7.50 4.
 3.25 2.10 6.17.25 2.50 5. 3.25 8.70 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 5.102. 7.20. 5. 2. 0. 2. 376.67 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 8.103.11. 7.15. 9. 0. 9. 377.50 1.70 7.
 2.25 2.30 5.16.25 1.20 4. 2.25 1.80 3.16.25 6.20 9. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 9.104.12. 7.15. 9. 0. 9. 380.00 0.60 8.
 2.25 0.9010. 2.25 0.80 9. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
10.105.13. 7. 5. 9. 0. 9. 382.50 0.5010.
 2.25 1.00 8. 2.25 0.50 9. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
11.106.14. 7. 5. 9. 0. 9. 385.00 2.1011.
 2.25 5.20 6.16.25 1.80 4. 2.25 2.20 7.16.25 1.50 9. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
12.107.15.20.15.11. 0.11. 387.50 5.20 6.
17.25 1.50 3. 3.25 1.60 4.17.25 0.70 5. 3.25 4.2011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
13.108.16.20.15.11. 0.11. 390.00 2.50 9.
 3.25 1.70 7. 3.25 1.50 8.17.25 0.9010. 3.25 1.1011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
14.109.17.20. 5.11. 0.11. 392.50 3.1012.
17.25 8.80 6.17.25 3.10 9. 3.25 2.1011. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
  .
  .
  .
10.180.13. 7. 5. 9. 0. 9. 670.50 0.5010.
 2.25 1.00 8. 2.25 0.50 9. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
```

```
11.181.14. 7. 5. 9. 0. 9. 673.00 2.1011.
 2.25 5.20 6.16.25 1.80 4. 2.25 2.20 7.16.25 1.50 9. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 5.182. 7.20. 5. 2. 0. 2. 676.67 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
12.183.15.20.15.11. 0.11. 687.50 5.20 6.
17.25 1.50 3. 3.25 1.60 4.17.25 0.70 5. 3.25 4.2011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
13.184.16.20.15.11. 0.11. 690.00 2.50 9.
 3.25 1.70 7. 3.25 1.50 8.17.25 0.9010. 3.25 1.1011. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
14.185.17.20. 5.11. 0.11. 692.50 3.1012.
17.25 8.80 6.17.25 3.10 9. 3.25 2.1011. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 5.186. 8.20. 5. 2. 0. 2. 693.33 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 3.187. 4.50. 5. 1. 0. 1. 695.00 7.90 6.
18.25 9.90 1. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 6.188. 9.20. 5. 2. 0. 2. 696.50 8.20 3.
17.25 9.70 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 5.189. 6.20. 5. 2. 0. 2. 710.00 1.10 1.
 3.25 7.90 6.17.2510.40 2. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0. 0.00 0.00 0.
```

Appendix R. *The QUEUES.OUT Output File*

This file is one of the output files created by the SLAM cargo route simulation found in Appendices D and E. An excerpt of the file created from the hypothetical twelve-airport route system follows:

BEGIN PORT	END PORT	STOP AT	PLANE NUMBER	ON FLIGHT	DEPART TIME	NUMBER OF	MEAN DELAY	TOTAL DELAY
1	4	1	2	191	722.50	16	7.80	124.80
1	4	1	2	241	902.50	23	7.80	179.40
1	4	1	2	286	1082.50	14	7.80	109.20
1	4	1	2	335	1262.50	13	7.80	101.40

SUBTOTAL DELAY FOR CARGO GOING FROM 1 TO 4 AT PORT 1 IS 514.80

TOTAL DELAY FOR CARGO GOING FROM 1 TO 4 IS 514.80

.
.
.

BEGIN PORT	END PORT	STOP AT	PLANE NUMBER	ON FLIGHT	DEPART TIME	NUMBER OF	MEAN DELAY	TOTAL DELAY
2	3	1	1	190	720.00	3	7.90	23.70
2	3	1	1	214	810.00	5	7.90	39.50
2	3	1	1	240	900.00	4	7.90	31.60
2	3	1	1	263	990.00	5	7.90	39.50
2	3	1	1	284	1080.00	5	7.90	39.50
2	3	1	1	309	1170.00	4	7.90	31.60
2	3	1	1	334	1260.00	7	7.90	55.30
2	3	1	1	358	1350.00	1	7.90	7.90

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 3 AT PORT 1 IS 268.60

2	3	2	5	164	607.50	3	112.50	337.5(
2	3	2	9	195	732.50	7	8.20	57.40
2	3	2	9	207	768.50	7	8.20	57.40
2	3	2	9	211	804.50	7	8.20	57.40
2	3	2	5	193	727.50	6	97.50	585.00
2	3	2	9	224	840.50	12	8.20	98.40
2	3	2	9	233	876.50	8	8.20	65.60
2	3	2	5	227	847.50	7	103.93	727.50
2	3	2	9	243	912.50	6	8.20	49.20
2	3	2	9	252	948.50	8	8.20	65.60
2	3	2	9	261	984.50	7	8.20	57.40
2	3	2	5	259	967.50	6	97.50	585.00

2	3	2	9	270	1020.50	8	8.20	65.60
2	3	2	9	282	1056.50	8	8.20	65.60
2	3	2	9	289	1092.50	8	8.20	65.60
2	3	2	9	301	1128.50	13	8.20	106.60
2	3	2	9	306	.164.50	7	8.20	57.40
2	3	2	5	288	1087.50	4	82.50	330.00
2	3	2	9	319	1200.50	7	8.20	57.40
2	3	2	9	328	1236.50	8	8.20	65.60
2	3	2	5	321	1207.50	7	52.50	367.50
2	3	2	9	338	1272.50	7	8.20	57.40
2	3	2	9	346	1308.50	13	8.20	106.60
2	3	2	9	355	1344.50	8	8.20	65.60
2	3	2	5	353	1327.50	1	22.50	22.50
2	3	2	9	365	1380.50	12	8.20	98.40

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 3 AT PORT 2 IS 4275.20

TOTAL DELAY FOR CARGO GOING FROM 2 TO 3 IS 4543.80

BEGIN PORT	END PORT	STOP AT	PLANE NUMBER	ON FLIGHT	DEPART TIME	NUMBER OF	MEAN DELAY	TOTAL DELAY
2	12	1	1	120	450.00	1	26.15	26.15
2	12	1	1	190	720.00	1	26.15	26.15
2	12	1	1	240	900.00	1	26.15	26.15
2	12	1	1	145	540.00	1	26.15	26.15
2	12	1	1	284	1080.00	3	26.15	78.45
2	12	1	1	169	630.00	1	26.15	26.15

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 1 IS 209.20

2	12	2	9	117	444.50	3	30.68	92.05
2	12	2	10	79	285.00	1	10.75	10.75
2	12	2	10	101	375.00	1	10.75	10.75
2	12	2	9	112	408.50	2	13.00	26.00
2	12	2	5	99	367.50	1	82.50	82.50
2	12	2	10	174	645.00	4	10.75	43.00
2	12	2	5	164	607.50	2	67.50	135.00
2	12	2	9	130	480.50	2	33.60	67.20
2	12	2	9	139	516.50	3	28.47	85.40
2	12	2	10	128	465.00	4	10.75	43.00
2	12	2	10	245	915.00	3	10.75	32.25
2	12	2	5	193	727.50	1	172.50	172.50
2	12	2	9	252	948.50	1	49.00	49.00
2	12	2	9	270	1020.50	1	49.00	49.00
2	12	2	9	149	552.50	3	22.95	68.85
2	12	2	5	132	487.50	1	52.50	52.50
2	12	2	9	166	624.50	2	30.20	60.40
2	12	2	10	290	1095.00	2	10.75	21.50
2	12	2	5	259	967.50	3	112.50	337.50

2	12	2	10	150	555.00	4	10.75	43.00
2	12	2	10	268	1005.00	3	10.75	32.25
2	12	2	9	306	1164.50	2	40.33	80.65
2	12	2	9	319	1200.50	1	13.00	13.00
2	12	2	9	176	660.50	2	51.60	103.20
2	12	2	9	188	696.50	3	18.20	54.60
2	12	2	9	338	1272.50	1	42.20	42.20
2	12	2	9	328	1236.50	1	49.65	49.65

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 2 IS 1857.70

2	12	3	15	118	474.70	2	18.85	37.70
2	12	3	11	103	421.50	2	83.75	167.50
2	12	3	1	120	476.15	2	59.10	118.20
2	12	3	1	190	746.15	1	59.10	59.10
2	12	3	15	134	534.70	3	18.85	56.55
2	12	3	1	240	926.15	1	59.10	59.10
2	12	3	11	253	997.50	1	35.50	35.50
2	12	3	11	119	493.50	1	71.75	71.75
2	12	3	11	272	1069.50	1	35.75	35.75
2	12	3	11	140	565.50	2	35.50	71.00
2	12	3	15	152	594.70	1	18.85	18.85
2	12	3	1	145	566.15	2	69.60	139.20
2	12	3	15	168	654.70	2	18.85	37.70
2	12	3	1	284	1106.15	3	59.10	177.30
2	12	3	1	309	1196.15	1	89.10	89.10
2	12	3	11	308	1213.50	2	71.75	143.50
2	12	3	15	183	714.70	4	18.85	75.40
2	12	3	11	178	709.50	1	35.50	35.50
2	12	3	1	169	656.15	1	88.85	88.85
2	12	3	15	341	1314.70	1	18.85	18.85
2	12	3	1	334	1286.15	1	59.10	59.10

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 3 IS 1595.50

2	12	4	15	118	493.55	2	3.95	7.90
2	12	4	10	79	295.75	1	203.15	203.15
2	12	4	10	101	385.75	1	113.15	113.15
2	12	4	10	174	655.75	4	143.15	572.60
2	12	4	15	134	553.55	3	3.95	11.85
2	12	4	10	128	475.75	4	83.15	332.60
2	12	4	10	245	925.75	3	53.15	159.45
2	12	4	15	152	613.55	1	3.95	3.95
2	12	4	15	168	673.55	2	3.95	7.90
2	12	4	10	290	1105.75	2	53.15	106.30
2	12	4	10	150	565.75	4	143.15	572.60
2	12	4	10	268	1015.75	3	303.15	909.45
2	12	4	15	183	733.55	4	3.95	15.80
2	12	4	15	341	1333.55	1	3.95	3.95

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 4 IS 3020.65

2	12	5	15	118	497.50	2	255.00	510.00
2	12	5	15	134	557.50	3	375.00	1125.00
2	12	5	15	152	617.50	1	495.00	495.00
2	12	5	15	168	677.50	2	435.00	870.00
2	12	5	15	183	737.50	4	555.00	2220.00
2	12	5	15	341	1337.50	1	15.00	15.00

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 5 IS 5235.00

2	12	6	17	123	498.90	2	6.35	12.70
2	12	6	17	204	798.90	4	6.35	25.40
2	12	6	17	137	558.90	4	6.35	25.40
2	12	6	17	249	978.90	3	6.35	19.05
2	12	6	17	298	1158.90	2	6.35	12.70
2	12	6	17	171	678.90	2	6.35	12.70
2	12	6	17	326	1278.90	1	6.35	6.35
2	12	6	17	185	738.90	2	6.35	12.70
2	12	6	17	343	1338.90	2	6.35	12.70

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 6 IS 139.70

2	12	7	16	170	640.70	2	18.75	37.50

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 7 IS 37.50

2	12	8	16	170	659.45	2	4.15	8.30

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 8 IS 8.30

2	12	9	17	123	505.25	5	247.25	1236.25
2	12	9	17	204	805.25	5	7.25	36.25
2	12	9	17	137	565.25	6	407.25	2443.50
2	12	9	17	249	985.25	4	7.25	29.00
2	12	9	14	276	1033.00	1	19.50	19.50
2	12	9	17	280	1105.25	1	7.25	7.25
2	12	9	14	162	601.00	2	511.50	1023.00
2	12	9	16	170	635.75	2	4.95	9.90
2	12	9	17	298	1165.25	5	7.25	36.25
2	12	9	17	171	685.25	2	547.25	1094.50
2	12	9	17	326	1285.25	4	7.25	29.00
2	12	9	14	200	745.00	2	547.50	1095.00
2	12	9	17	185	745.25	2	547.25	1094.50
2	12	9	17	343	1345.25	3	7.25	21.75

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 9 IS 8175.65

2	12	10	16	170	663.60	2	448.90	897.80

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 10 IS 897.80

2	12	11	17	204	752.50	7	3.10	21.70
2	12	11	17	217	812.50	5	3.10	15.50
2	12	11	17	249	932.50	7	3.10	21.70
2	12	11	17	265	992.50	4	3.10	12.40
2	12	11	17	280	1052.50	3	3.10	9.30
2	12	11	17	298	1112.50	8	3.10	24.80
2	12	11	17	312	1172.50	5	3.10	15.50
2	12	11	17	326	1232.50	2	3.10	6.20
2	12	11	17	343	1292.50	12	3.10	37.20
2	12	11	17	360	1352.50	4	3.10	12.40

SUBTOTAL DELAY FOR CARGO GOING FROM 2 TO 12 AT PORT 11 IS 176.70

TOTAL DELAY FOR CARGO GOING FROM 2 TO 12 IS 21353.70

.
.
.

TOTAL DELAY FOR ALL CARGO IS 222647.22

Appendix S. *The LEGS.OUT Output File*

This file is one of the output files created by the SLAM cargo route simulation found in Appendices D and E. An excerpt of the file created from the hypothetical twelve-airport route system follows:

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
190	1	1	1	1	12	26	720.00
190	1	1	1	1	9	20	720.00
190	1	1	1	2	3	3	720.00
190	1	1	1	2	12	1	720.00
190	CAPACITY[50]		1		UNUSED:	0	
190	1	1	2	1	12	26	720.00
190	1	1	2	1	9	20	720.00
190	1	1	2	2	12	1	720.00
190	1	1	2	2	7	1	746.15
190	1	1	2	2	8	1	746.15
190	1	1	2	2	12	1	746.15
190	CAPACITY[50]		2		UNUSED:	0	
190	1	1	3	8	2	6	770.60
190	1	1	3	9	1	8	770.60
190	1	1	3	7	5	1	770.60
190	1	1	3	8	5	2	770.60
190	1	1	3	10	1	1	770.60
190	1	1	3	10	3	1	770.60
190	CAPACITY[50]		3		UNUSED:	31	
190	1	1	4	8	2	6	770.60
190	1	1	4	9	1	8	770.60
190	1	1	4	10	1	1	770.60
190	1	1	4	10	3	1	770.60
190	1	1	4	5	2	3	792.55
190	CAPACITY[50]		4		UNUSED:	31	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
191	2	2	1	1	12	14	722.50
191	2	2	1	1	9	11	722.50
191	2	2	1	1	4	16	722.50
191	2	2	1	6	4	6	722.50
191	2	2	1	9	4	1	722.50

191	2	2	1	8	4	1	722.50
191	2	2	1	5	4	1	722.50
191	CAPACITY[50]		1		UNUSED:	0	
191	2	2	2	1	12	14	722.50
191	2	2	2	1	9	11	722.50
191	CAPACITY[50]		2		UNUSED:	25	
191	2	2	3	10	4	2	770.80
191	2	2	3	8	4	1	770.80
191	2	2	3	9	4	1	770.80
191	CAPACITY[50]		3		UNUSED:	46	
191	2	2	4	4	1	12	793.35
191	CAPACITY[50]		4		UNUSED:	38	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
192	3	3	1	1	6	50	725.00
192	CAPACITY[50]		1		UNUSED:	0	
192	3	3	2	6	1	27	751.15
192	3	3	2	6	4	1	751.15
192	CAPACITY[50]		2		UNUSED:	22	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
193	5	4	1	2	3	6	727.50
193	5	4	1	2	7	3	727.50
193	5	4	1	2	8	4	727.50
193	5	4	1	2	5	5	727.50
193	5	4	1	2	12	2	727.50
193	CAPACITY[20]		1		UNUSED:	0	
193	5	4	2	8	2	12	731.85
193	5	4	2	7	2	4	731.85
193	5	4	2	5	2	4	731.85
193	CAPACITY[20]		2		UNUSED:	0	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
194	6	5	1	6	1	19	730.00
194	6	5	1	10	1	1	730.00
194	CAPACITY[20]		1		UNUSED:	0	

194	6	5	2	1	6	13	734.35
194	6	5	2	2	7	3	734.35
194	6	5	2	2	8	4	734.35
194	CAPACITY[20]		2		UNUSED:	0	
194	6	5	3	6	1	14	759.50
194	6	5	3	10	1	2	759.50
194	CAPACITY[20]		3		UNUSED:	4	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
195	9	6	1	2	12	3	732.50
195	9	6	1	2	3	7	732.50
195	9	6	1	2	8	5	732.50
195	9	6	1	2	7	4	732.50
195	9	6	1	2	10	1	732.50
195	CAPACITY[20]		1		UNUSED:	0	
195	9	6	2	3	2	11	757.95
195	CAPACITY[20]		2		UNUSED:	9	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
196	10	7	1	2	8	7	735.00
196	10	7	1	2	12	4	735.00
196	10	7	1	2	7	3	735.00
196	10	7	1	2	5	5	735.00
196	10	7	1	2	10	1	735.00
196	CAPACITY[20]		1		UNUSED:	0	
196	10	7	2	2	8	7	735.00
196	10	7	2	2	12	4	735.00
196	10	7	2	2	7	3	735.00
196	10	7	2	2	10	1	735.00
196	10	7	2	3	6	5	745.75
196	CAPACITY[20]		2		UNUSED:	0	
196	10	7	3	12	5	1	765.10
196	10	7	3	2	5	1	765.10
196	CAPACITY[20]		3		UNUSED:	18	
196	10	7	4	5	2	6	770.85
196	10	7	4	7	2	4	770.85
196	CAPACITY[20]		4		UNUSED:	10	

FLIGHT	TAIL	MISSON	LEG	CARGO	CARGO	NUMBER	TIME

NUMBER	NUMBER	NUMBER	NUMBER	ORIG	DEST	OF	GOT ON
197	11	8	1	8	7	1	737.50
197	11	8	1	2	7	5	737.50
197	11	8	1	7	5	1	737.50
197	CAPACITY[7]		1		UNUSED:	0	
197	11	8	2	7	5	1	737.50
197	11	8	2	7	2	4	741.45
197	11	8	2	7	5	2	741.45
197	CAPACITY[7]		2		UNUSED:	0	
197	11	8	3	5	4	1	760.00
197	CAPACITY[7]		3		UNUSED:	6	
197	11	8	4	8	2	3	763.45
197	CAPACITY[7]		4		UNUSED:	4	
197	11	8	5	3	6	2	781.50
197	11	8	5	2	8	2	781.50
197	11	8	5	2	10	1	781.50
197	11	8	5	2	7	1	781.50
197	11	8	5	2	12	1	781.50
197	CAPACITY[7]		5		UNUSED:	0	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
198	12	9	1	6	10	1	740.00
198	12	9	1	2	8	5	740.00
198	12	9	1	2	10	1	740.00
198	CAPACITY[7]		1		UNUSED:	0	
198	12	9	2	6	10	1	740.00
198	12	9	2	2	10	1	740.00
198	12	9	2	8	2	3	742.85
198	12	9	2	8	5	2	742.85
198	CAPACITY[7]		2		UNUSED:	0	
198	12	9	3	8	2	3	742.85
198	12	9	3	8	5	2	742.85
198	12	9	3	10	1	1	746.00
198	CAPACITY[7]		3		UNUSED:	1	

FLIGHT NUMBER	TAIL NUMBER	MISSON NUMBER	LEG NUMBER	CARGO ORIG	CARGO DEST	NUMBER OF	TIME GOT ON
199	13	10	1	2	8	5	742.50
199	13	10	1	2	10	1	742.50

```
199      13      10      1      6      10      1    742.50
199    CAPACITY[ 7]      1             UNUSED:  0

199      13      10      2      2       8      5    742.50
199      13      10      2     10       6      1    745.25
199      13      10      2     10       3      1    745.25
199    CAPACITY[ 7]      2             UNUSED:  0

199      13      10      3     10       6      1    745.25
199      13      10      3     10       3      1    745.25
199      13      10      3      8       2      3    748.50
199      13      10      3      8       7      1    748.50
199    CAPACITY[ 7]      3             UNUSED:  1
```

etc.

Bibliography

1. Ackley, M. and others. "Optimization Applications at the Military Airlift Command: Importance and Difficulties," Unpublished Paper Presented to the Minisymposia on Large Scale Optimization via Simplex and Interior Point: Algorithms and Applications at the Second International Conference on Industrial and Applied Mathematics, Washington, D.C. (July 8–12, 1991).
2. Baker, K. R. *Introduction to Sequencing and Scheduling.* New York: John Wiley & Sons, Inc., 1974.
3. Bodin, Lawrence D. "Twenty Years of Routing and Scheduling," *Operations Research, 38:* 571-579 (July/August 1990).
4. Bodin, Lawrence D. and others. "Routing and Scheduling of Vehicles and Crews: The State of the Art," *Computers and Operations Research, 10:* 62–212 (1983).
5. Bookbinder, James H. and Steven H. Edwards. "School Bus Routing for Program Scheduling," *Computers and Operations Research, 17:* 79–94 (January/February 1990).
6. Borsi, Captain John J. Discussions. Air Force Institute of Technology, Wright-Patterson AFB, OH. July 1991 through February 1992.
7. Byong-Hun Ahn and Jae-Yeong Shin. "Vehicle-routing with Time Windows and Time-varying Congestion," *Journal of the Operational Research Society, 42:* 393–400 (May 1991).
8. Deal, Don E. and others. "An Expert System Scheduler: Some Reflections on Expert Systems Development," *Computers and Operations Research, 17:* 571–580 (June 1990).
9. Dobson, Gregory and Uday S. Karmarkar. "Simultaneous Resource Scheduling to Minimize Weighted Flow Times," *Operations Research, 37:* 592–600 (July–August 1989).
10. Dodin, Bajis and K. Hung Chan. "Application of Production Scheduling Methods to External and Internal Audit Scheduling," *European Journal of Operational Research, 52:* 267–279 (June 1991).
11. Ferland, Jacques A. and Luc Fortin. "Vehicle Scheduling with Sliding Time Windows," *European Journal of Operational Research, 38:* 213–226 (January 1989).
12. Fisher, Marshall L. and Moshe B. Rosenwein. "An Interactive Optimization System for Bulk Cargo Ship Scheduling," *Naval Research Logistics, 36:* 27–42 (January 1989).

13. Hillier, Fredrick S., and Gerald J. Lieberman. *Introduction to Mathematical Programming.* New York: McGraw-Hill Publishing Company, 1990.

14. Kikuchi, Shinya and Jong-ho Rhee. "Scheduling Method for Demand-Responsive Transportaion System," *Journal of Transportation Engineering, 115:* 630-645 (November 1989).

15. Litko, Lieutenant Colonel Joseph R., Force Structure Analysis, Command Analysis Group, Military Airlift Command. Telephone Interviews. Wright-Patterson AFB OH, 8 August through 16 December 1991.

    1989).

16. Nemhauser, George L. and Laurence A. Wolsey. *Integer and Combinatorial Optimization.* New York: John Wiley & Sons, Inc., 1988.

17. Petersen, A. R. and A. J. Taylor. "An Optimal Scheduling System for the Welland Canal," *Transportation Science, 22:* 173-185 (August 1988).

18. Pritsker, A. Alan B. *Introduction to Simulation and SLAM II* (Third Edition). New York: Halsted Press, John Wiley & Sons, Inc., and Systems Publishing Corp., 1986.

19. Schwartz, Laurence. "Enhancing Aircraft Utilization: Improved Use of Cargo-Holding Time," Report AC001R2 prepared persuant to DOD contract MDA903-90-C-0006. Bethesda, MD: Logistics Management Institute, September 1990.

20. Sheppard. "Peacetime Airlift: Job # 1, Too !" *Defense Transportation Journal, 46. Number 5:* 11-20 (1990).

21. Solomon, Marius M. and Jacques Desrosiers. "Time Window Constrained Routing and Scheduling Problems," *Transportation Science, 22:* 1-13 (February 1988).

22. Teodoric, Dusan and Emina Krcmar-Nozic. "Multicriteria Model to Determine Flight Frequencies on an Airline Nework under Competitive Conditions," *Transportation Science, 23:* 14-25 (February 1989).

Vita

Captain Justin E. Moul was born on 18 October 1960 in York, Pennsylvania. He graduated from Westmont Hilltop High Schoool, in Johnstown, Pennsylvania, in 1978 and attended Indiana University in Bloomington, Indiana. He received the degree of Bachelor of Science in Accounting, with Honors, in August 1982, and the degree of Master of Business Administration in Finance and Management Information Systems in August 1983. Upon graduation, he received his Air Force commission through the AFROTC program. After completing Initial Qualification Training for the Minuteman II Intercontinental Ballistic Missile System, Captain Moul was assigned to Whiteman AFB, Missouri, in January 1984. As a Missile Combat Crewmember at Whiteman, Captain Moul served as Deputy Commander and Commander for the 509th Strategic Missile Squadron; and as Deputy Commander, Senior Deputy Commander, and Commander for the 351st Strategic Missile Wing Standardization/Evaluation Division. After completing his crew tour in 1988, Captain Moul remained at Whiteman where he served as Wing Scheduling Officer; Chief, Operations Documentation Branch, and Emergency War Order Planner/Instructor. In August 1990, Captain Moul was re-assigned to the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Permanent address: 339 Tioga Street
Johnstown, Pennsylvania
15905

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1992	Master's Thesis

4. TITLE AND SUBTITLE
A METHOD FOR DETERMINING SCHEDULE DELAY INFORMATION IN A CHANNEL CARGO ROUTE NETWORK

5. FUNDING NUMBERS

6. AUTHOR(S)
Justin E. Moul, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GST/ENS/92M-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This research develops a method for measuring schedule effectiveness by determining the amounts of enroute cargo delay caused by a given aircraft mission schedule. The method is designed to generate information which helps identify flights for which different scheduling might decrease overall cargo delay in the entire network, given that all non-scheduling factors are held constant. The research uses a simplified twelve-airport cargo route network to test the methodology.

14. SUBJECT TERMS
Simulation, Scheduling

15. NUMBER OF PAGES
143

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL